


For Reference

NOT TO BE TAKEN FROM THIS ROOM

Ex LIBRIS
UNIVERSITATIS
ALBERTAENSIS





Digitized by the Internet Archive
in 2023 with funding from
University of Alberta Libraries

<https://archive.org/details/Erickson1971>

THE UNIVERSITY OF ALBERTA

A METHODOLOGY FOR EVALUATING MAN-COMPUTER SYSTEMS

by



Ronald W. Erickson

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE
OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTING SCIENCE

EDMONTON, ALBERTA

Spring, 1971

ABSTRACT

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled A Methodology for Evaluating Man-Computer Systems submitted by Ronald W. Erickson in partial fulfillment of the requirements for the degree of Master of Science.

ABSTRACT

Enough data are currently available to support the view that the man and the computer can be highly sophisticated central processors. But when linked together for interaction, they are joined by somewhat limited, and even primitive, communication channels.

This research focuses on the question of optimal interaction between man and machine. Somehow, a Man-computer (M-C) system must not only offer a user equipment components capable of performing his application in an acceptable fashion, but also offer a user a system which is practical to employ, in terms of cost.

In order to analyse the performance of an M-C system, those variables and characteristics of a system which affect the flow of useful information in the M-C closed loop must be identified. With the trend to greater complexity in computing systems, plus the proliferating number of terminal devices now available, these variables are difficult to isolate. Usually, a designer is compelled to build systems with little knowledge of how a user will react to a particular group of equipments after they are assembled.

In this study, the computer side of the interface is considered as being composed of three separate subsystems: Input, Computer and Output. The methodology advanced considers those characteristics within the definition of six design parameters. An analytical method whereby the design parameters may be measured and used to calculate a subsystem's worthiness to a user is described.

ACKNOWLEDGMENTS

The successful completion of this research project must be attributed to several persons. Foremost, I wish to thank Dr. J. Tartar for his continual encouragement and helpful direction given throughout this study. Also, appreciation must go to my wife, Sharon, for her sincere understanding and perseverance during the past year.

Finally, I am grateful to Dr. D. B. Scott for allocating the financial support received for this research.

TABLE OF CONTENTS

	Page
CHAPTER I: INTRODUCTION	
1.1 Studies of Man-Computer Systems	1
1.2 Need for Research in this Area	1
1.3 Methods of Evaluating M-C Systems	2
1.4 Evaluation Approach Used in this Study	3
CHAPTER II: BACKGROUND RESEARCH	
2.1 Overview of Background Required	5
2.2 Man-Computer Systems	5
2.2.1 The M-C Interface	7
2.2.2 Models Used in Man-Computer Study	11
2.3 The Evaluation Problem	12
2.4 Approaches Taken Towards Evaluation	15
2.5 Structure of Techniques for Evaluation	19
2.5.1 Need for a Methodology	21
2.5.2 Design and Validation of Methodologies	23
2.6 Subdividing the Computer-environment	24
2.7 Selection of Design Parameters for Computer- environment	27
CHAPTER III: THE METHODOLOGY	
3.1 Overview of the Methodology	29
3.2 Model of the Computer-environment	31

	Page
3.2.1 Function of the Input Subsystem	33
3.2.2 Function of the Computer Subsystem	38
3.2.3 Function of the Output Subsystem	40
3.2.4 The External Environment	41
3.3 The Worthiness Index	42
3.3.1 Concept of a Matched Subsystem	45
3.3.2 Response Time and Configuration of the Computer Subsystem	49
3.4 Concept of Requirements Space	51
3.4.1 Requirements to a Subsystem	54
3.4.2 Functional Requirements at the Interface	57
3.5 The Design Parameters	60
3.5.1 Relative Importances Between DP's	65
3.5.2 Price Functions of DP's	66
3.6 Evaluating a Subsystem	71
3.7 Limitations of the Methodology	77

CHAPTER IV: THE VALIDATION

4.1 Validation and Application of the Methodology ...	78
4.2 Requirements of an Experimental Setting	79
4.3 The Experimental Setting	80
4.3.1 Description of the Test Computer Subsystem	81
4.3.2 The Data Base Used for Evaluation	82
4.4 Choosing Characteristic Equations	84
4.5 The Experimental Procedure	89
4.6 Results of the Experiment	93

	Page
CHAPTER V: CONCLUSIONS	
5.1 Summary and Conclusions	103
5.1.1 Development of the Methodology	103
5.1.2 Validation of the Methodology	104
5.1.3 Uses of the Experimental Setting	106
5.2 Directions for Further Research	106
BIBLIOGRAPHY	108
APPENDIX A: DESCRIPTION OF THE SIMULATOR MONITOR	111
APPENDIX B: OPERATING THE EXPERIMENTAL SETTING	132
APPENDIX C: SAMPLE CALCULATION OF WORTHINESS	135

LIST OF FIGURES

	Page
Fig. 2-1 Man-computer system communication link	6
Fig. 2-2 Possible Interface positions from Information and Decision Theory viewpoint	8
Fig. 2-3 Elements of information transfer process	9
Fig. 2-4 Schematic TSS information flow	10
Fig. 2-5 Environments of a Man-computer system	13
Fig. 2-6 Two critical links of Man-computer system	13
Fig. 2-7 Model of computer system usage	18
Fig. 2-8 General display system	22
Fig. 2-9 Principle equipment groups of Computer- environment	25
Fig. 3-1 Principle operations required by evaluation technique	30
Fig. 3-2a Main equipment groups of M-C system	32
Fig. 3-2b Equipment groups considered in this study	32
Fig. 3-3 Model of Computer-environment used	34
Fig. 3-4 The Input subsystem	35
Fig. 3-5 GRID system as Input subsystem	37
Fig. 3-6 Computer subsystem showing External environment	39
Fig. 3-7 The Output subsystem	39
Fig. 3-8 A process control situation shown in relation to M-C closed loop	43
Fig. 3-9 Worthiness vs Cost for various hypothetical configurations of equipments	46

	Page
Fig. 3-10 The electrical impedance analogue for matched systems	46
Fig. 3-11 Summing Junction Model to analyse capabilities vs requirements	47
Fig. 3-12 Hypothetical curves resulting from varying configurations on two requirement sets	47
Fig. 3-13 C-W graph showing different response times for configurations	50
Fig. 3-14 R - Space consisting of three subsets	53
Fig. 3-15 Functional and design requirements of subsystem	55
Fig. 3-16 Computer-environment characteristics	62
Fig. 3-17a Design parameters speed vs cost graph	67
Fig. 3-17b Speed vs equipment cost for storage	67
Fig. 3-18 Manufacturer and user value curves	70
Fig. 3-19 Hypothetical price function	70
Fig. 3-20 Iterative process for evaluating a subsystem .	73
Fig. 3-21 Worthiness vs cost graph of example problem ..	76
Fig. 4-1 R - Space used by experimental setting	83
Fig. 4-2 Flow chart of operations in validation test ..	90
Fig. 4-3 Program relationships in experiment	92
Fig. 4-4 Worthiness vs cost results for application 1 ..	100
Fig. 4-5 Worthiness vs cost results for application 2 ..	101
Fig. 4-6 Worthiness vs cost results for application 3 ..	102

LIST OF TABLES

	Page
Table 3-1 Flip-flop costs decrease with technical advancement	68
Table 3-2 Worthiness indices resulting from three sub-system configurations	76
Table 4-1 Data base elements considered in validation experiment	85
Table 4-2 Utilizations available from Nielsen's model .	88
Table 4-3 Relative costs of equipments used in experiment for configurations	95
Table 4-4 Experimental results for application 1	97
Table 4-5 Experimental results for application 2	98
Table 4-6 Experimental results for application 3	99

CHAPTER I

INTRODUCTION

1.1 Studies of Man-Computer Systems

There are now many kinds of computer systems but the focus of this research is the so-called on-line, real-time computing system. There is currently much interest in extending the computer capabilities to aid man in his creative, problem solving, designing functions; the consequence of the rapidly developing technology for this kind of data processing system raises questions related to optimal interaction between man and machine. Studies of this nature concerned with the application of biological and engineering data to problems relating to the mutual adjustment of man and machine have recently been labelled as ergonomic research.

Sackman [35] has defined man-machine digital systems as:

'... an evolving organization of people, computers and other equipment, including associated communication and support systems, and their integrated operation to regulate and control selected environment events to achieve system objectives'.

This complex group of interdependent entities suggests the complexity of this ergonomics problem.

1.2 Need for Research in this Area

The grossness of the problem of defining 'optimal' interaction between man and machine involves the selecting of optimal equipment configurations by designers. It appears that recognition of this problem has been made in current literature on ergonomics; however,

concrete solutions to the problem are lacking.

Theoretically, an optimal configuration consisting of 'optimal' components can be found by considering all variables affecting the overall effectiveness of the Man-computer (M-C) interface. But the biggest stumbling blocks thus far are defining these variables, their interrelations and establishing standardized weighting factors for each. A large portion of this study was directed towards describing a classification of variables which affect a system's value to a user.

In view of the high level of effort being allocated to the design and implementation of components of M-C systems, it is apparent that more attention should be made to the overall 'value' of interaction resulting from various configurations. In support of this statement, a recent review of ergonomics [34] states that:

'... more applied studies are needed to optimize the design of equipment; many costly devices are already on the market, but their ergonomic aspects are not always well defined. It seems that the rate of technical advancement is outstripping our capacity to evaluate its fruits at a desirable level of thoroughness'.

1.3 Methods of Evaluating M-C Systems

Any study or evaluation of a M-C system involves an analysis design parameters particular to a subject system. These parameters conceivably may be obtained by at least three techniques: simulation, analytical modelling and measurement.

Simulation often involves the running of a computer program which attempts to logically represent the dynamics of a real configuration in an experimental setting. Collecting statistics to obtain the parameters is straight forward; the accuracy of the data, however,

is determined by the closeness with which the program models the real system.

Analytical models, by their rigorous mathematical nature, are limited in capability to represent a real environment. This arises from the usual requirement of using manageable and exact mathematical forms to represent system variables.

The final technique involves direct measurement of operating characteristics. Inaccuracies through this technique to exist, these being mainly created by the inability to monitor all pertinent variables using hardware techniques and the bias introduced by software techniques. Often a combination of both has merits.

Before any attempt can be made to evaluate a M-C system, two prerequisites must be met:

- (a) a methodology with which to evaluate must be designed.

- (b) an application must exist with which to validate (a).

The first of these consists of a well defined set of rules which, when applied, will result in an evaluation of a system. The second provides for acceptable conditions or applications for method testing.

1.4 Evaluation Approach Used in this Study

The research reported here is focused on the development of a general methodology for evaluating the computer side of the M-C interface. It is based on a study of the principle characteristics and design of present Man-computer systems; more specifically, those characteristics which primarily determine the design parameters of accuracy, facility, flexibility, response, reliability and cost.

As is detailed in Chapter III, the computer side of the interface was analysed as three separate subsystems: Input, Computer and Output.

The methodology developed was tested by investigating the operating dynamics of a simulated time-sharing system used to represent the Computer subsystem with a single on-line user. The calculation of a Worthiness index is demonstrated and configuration curves of Worthiness versus cost were drawn to illustrate areas of optimal worth to a user.

The second chapter presents a discussion of the theories and research relevant to an understanding of digital man-machine systems. Also, the problems of evaluation are introduced. Chapter III presents a model of the computer side of the interface as well as the development and use of the methodology proposed. The experimental design for validating the methodology and the hypothesis used for the Computer subsystem are described in Chapter IV. Chapter V contains a discussion of the testing of the methodology and suggests possible extensions to this work.

CHAPTER II

BACKGROUND RESEARCH

2.1 Overview of Background Required

It is the intent of this chapter to describe a framework upon which the evaluation technique proposed in Chapter III was based. Research was undertaken to investigate the many orientations of study that have been, or are currently being, done in digital Man-machine systems. Attention was particularly directed towards definitions and concepts used by various researchers in Man-computer (M-C) studies. Those areas considered by researchers as 'critical bottlenecks' in the 'closed-loop' process are emphasized and several model concepts used are described. An effort was made to point out the many disciplines involved in these studies, each exhibiting different interests and directions. In essence, this chapter deals with the background felt necessary to design a reasonable methodology to evaluate M-C systems. The need for, the validation of, and the difficulty of designing such a technique are emphasized.

2.2 Man-computer Systems

The systems discussed in this study are those enabling man to communicate directly with the computer, and the computer to communicate with man. This action is illustrated in Fig. 2-1. The degree of complexity of this two-way communication varies greatly and is dependent on the nature of the application being carried out.

There is a wide variety of reasons for such communication bet-



Fig. 2-1. Man-computer system communication link

ween man and machine. It is generally agreed that man's information processing capability may be extended by employing a computer as a tool. In this way, a manmade work load may be shared by two processors, one being the human processor and the other being his 'assistant', the computer. The nature of this division of labour has been described by various researchers [3] [25]. It was suggested in Digital Computer Graphics [10] that '... (this) relationship must be a complementary one with each (processor) assisting the other in the things each does best'. In projecting his ideas on how this division should be proportioned, Martin [24] states that:

' ... there are certain thought processes which always remain in the domain of the human rather than machine activity. Others, however, are better done by the machine: when the two can be effectively combined, the result is more powerful than either process on its own'.

Man-computer systems vary along a broad spectrum of problems they are designed to solve; some are general purpose systems with 'flexibility' as a basic design requirement, while others are more specifically oriented such as 'dedicated' and 'common data base' systems. Examples of the latter are airline passenger scheduling systems, such as the PANAMAC System [23]. These dedicated systems

are more restricted and are designed to meet a specialized demand; they involve a rigidly prescribed interaction in contrast to general purpose systems. An example of the former system, incorporating a choice of versatile languages, communication devices and operating procedures, would be that offered to users at the University of Alberta under the Michigan Terminal System (MTS).

2.2.1 The M-C Interface

Man-computer interaction basically involves 'communication, elaboration and exchange' between two information structures: the human operator and the computer system. According to Carbonell [3], there must exist an 'evaluation, decision, and control model' directing the interchange. This task may be done by both processors, and it is the proportioning of division which establishes the position of this imaginary, but common, boundary called the Interface. Fig. 2-2 illustrates possible positions of the Interface, the location of which must be implicitly chosen by a user/designer according to his application.

Carbonell categorizes scientific and engineering conversational on-line usage of a time-sharing console as an example of the first case (Interface A), along with on-line debugging, conversational informational retrieval, etc. Programs in which the computer is in control and makes the decisions, such as in process control, are examples of the other extreme case (Interface C). Possibly a military application such as SAGE [35] could represent the intermediate case where all types of decision exist.

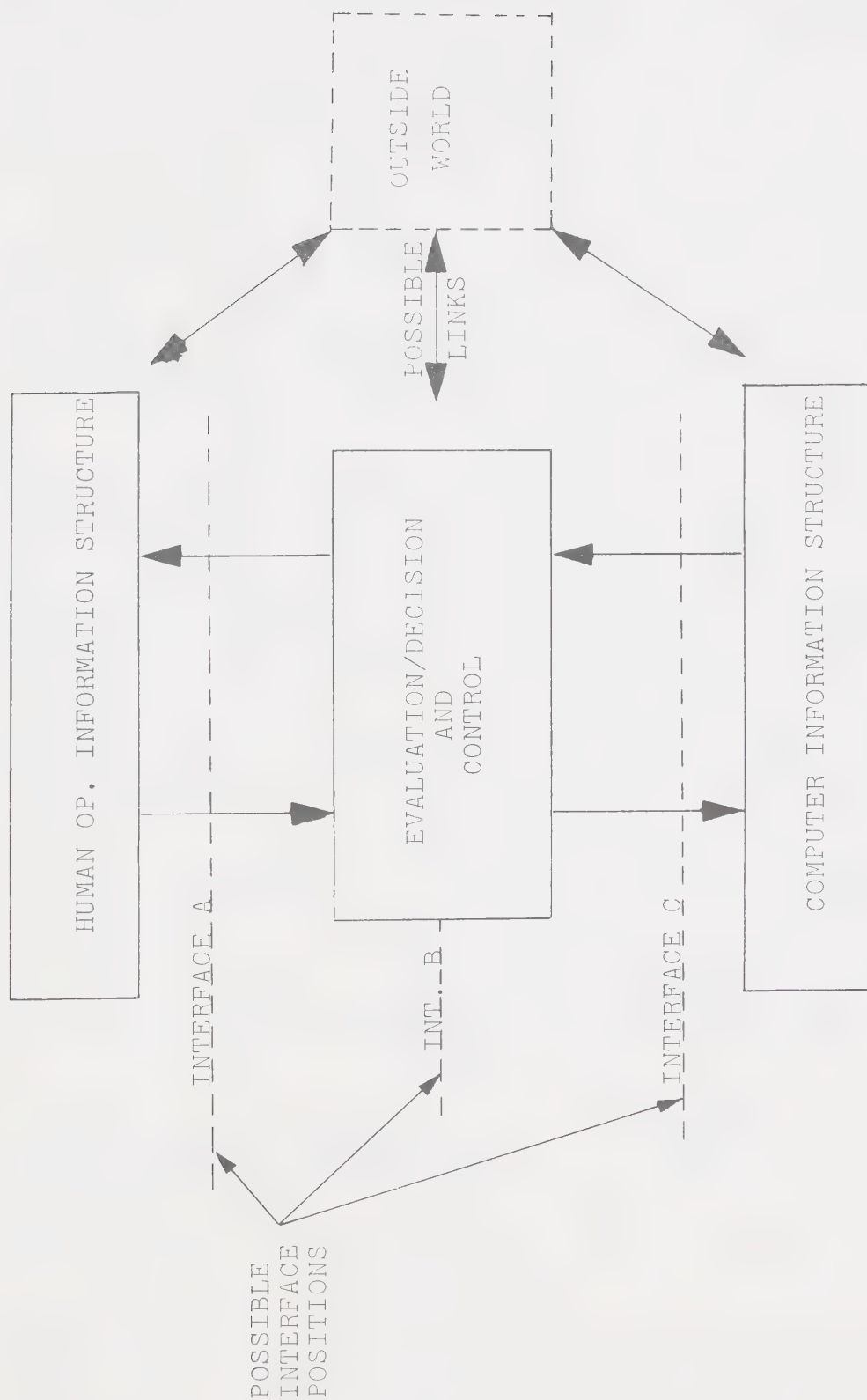


Fig. 2-2. Possible Interface positions from Information and Decision Theory viewpoint

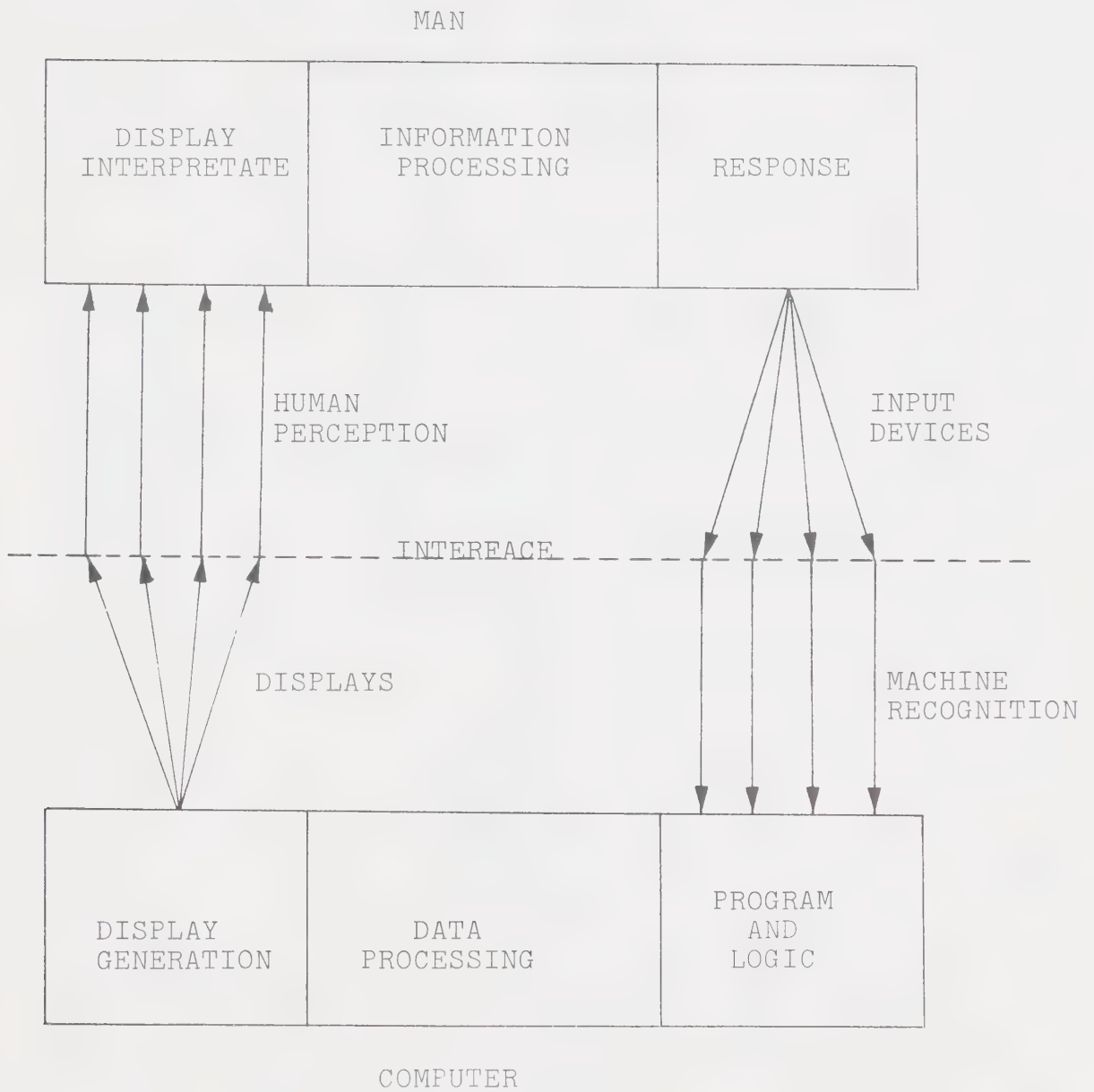


Fig. 2-3. Elements of information transfer process (Rosa)

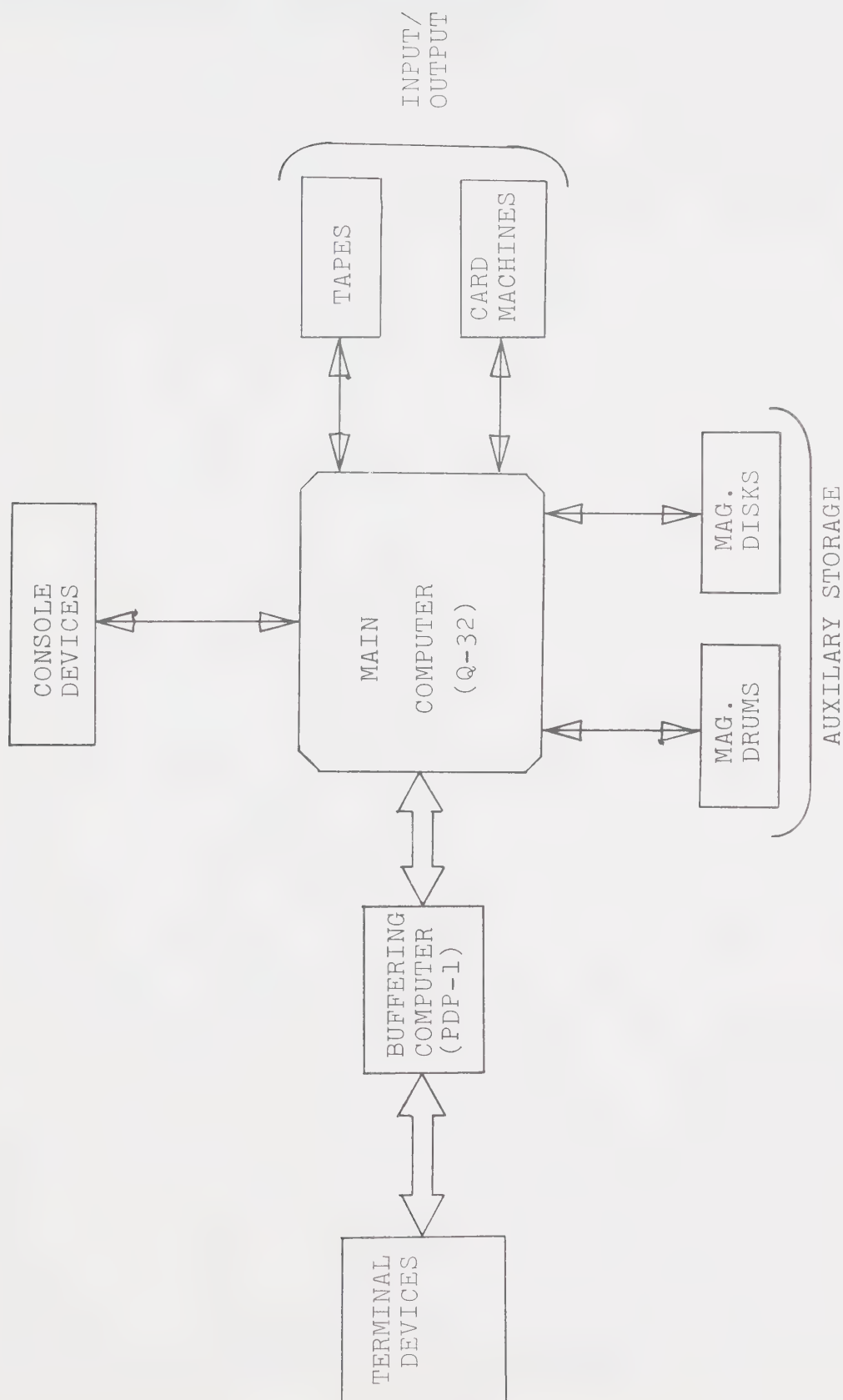


Fig. 2-4. Schematic TSS information flow

2.2.2 Models Used in Man-computer Study

In order to establish meaningful frames of reference about which to direct discussion of these systems, or parts of them, various models have been developed. Because of the many disciplines involved, ranging from human behavior research to computer system design, both the level and type of detail modelled is varied. The particular aspects of the system exhibited are, of course, related to the goals of an individual researcher. For example, Rosa [32], a behaviourist, considered the model illustrated in Fig. 2-3 to represent the elements involved with the information transfer problem.

More specific models dealing with only portions of the M-C loop, on either side of the Interface, are most commonly of use for specific areas of study. For example, on the computer side of the Interface, a system designer may be interested in models related to the information flow through actual equipments, both hardware and software. A model, such as that describing the SDC Experimental Time Sharing System [35], depicts the equipment groups used in this application (Fig. 2-4). In general, the grossness, or fineness, required of a model is a function of the degree of analysis required of the system.

Henceforth in this study, the man-side of the Interface is referred to as the Man-environment. The other side of the Interface, which includes all hardware and software seen by the user as aiding him in solving his application problem, is called the Computer-environment. Their relationship is illustrated in Fig. 2-5. The information lines in this illustration marked 'external events' refer to all inputs and outputs to the processors not directly considered as part of the M-C

closed loop; however, these events do affect the loop information flow. The external events of the Computer-environment are further described in Section 3.2.4.

The various disciplines involved with M-C study are usually concerned with one environment or the other. For example, Human Engineering is concerned with the Man-environment; computer system designing is normally associated with the Computer-environment.

2.3 The Evaluation Problem

As was pointed out in Section 2.1, a M-C system involves a closed loop information flow as illustrated in Fig. 2-6. According to Baker [1],

'One need work only a brief time in an operational on-line M-C system before one realizes that the man's role is performed in semi-detachment from the computer system for which he is providing direction. The daily business of the partnership is conducted through hardware mediators ... ',

whether they be elements of the input link or the output link as shown in Fig. 2-6. Metaphorically, there is no 'eyeball-to-eyeball' contact. Therefore, the critical feature affecting the man, the computer and the combined function of the two is the adequacy of the communication links between them. For most purposes, the evaluation problem involves minimizing the mismatch between the man and the computer caused by the 'noisy' communication links. A designer must attempt to select an 'optimal' configuration of equipments which best suits a particular user's application problem. Furthermore, to make the solution a practical one, the evaluation problem should include the selection of equipments which are economically feasible, i.e., those which a user can afford to use.

The design of a smooth Man-machine interface is not an easy task.

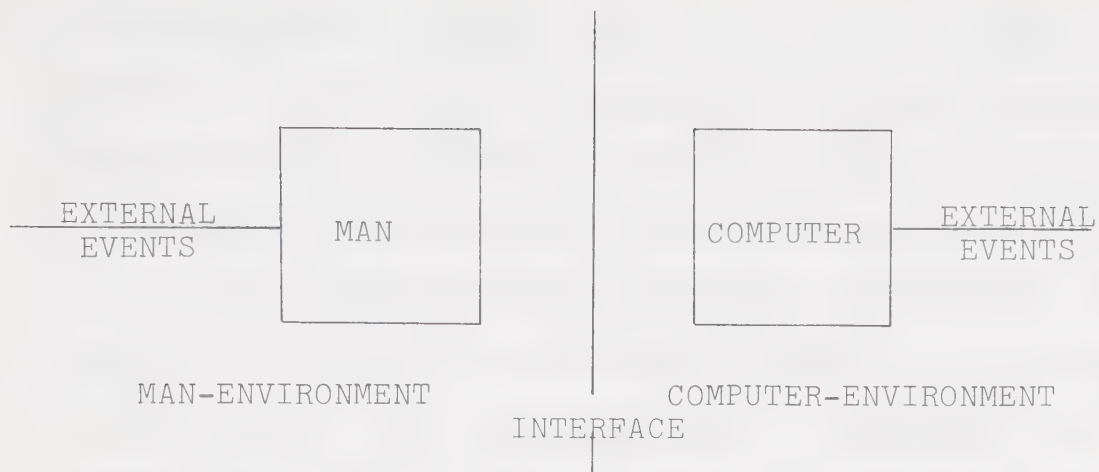


Fig. 2-5. Environments of a Man-computer system

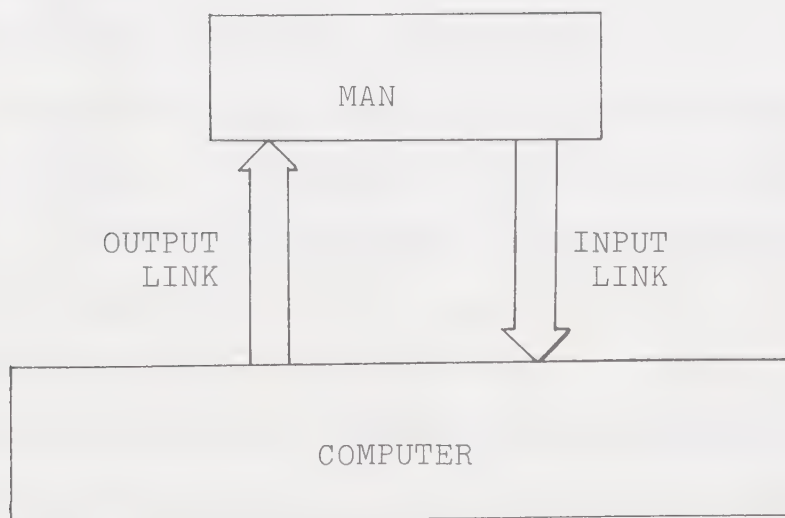


Fig. 2-6. Two critical links of Man-computer system

The problem invariably involves complex interrelations amongst variables from both the Man and the Computer-environments. For purposes of emphasizing the magnitude of the evaluation problem, the following examples illustrate variables of interest in the Human Factor's domain.

In a clear account of time-sharing system design problems, Nickerson [27] discussed the need for ergonomic research on such questions as programming and command languages, human job-swapping and system response times. Short response times are desirable for the user, but are costly and limit the number of simultaneous users. Is minimization of response time the main aim; do people have a minimum tolerable job-swap time and how does swapping affect performance? 'Ease of use' is a complex design objective since novice and expert users have conflicting needs. The advantage of easy access to the computer can lead to high loading and longer average response times, thus often creating user dissatisfaction.

Human Factor engineers are also interested in how a user's strategy is affected by computer system characteristics such as the 'system's accessibility, responsiveness, predictability, charging policy, etc.' [27]. Nickerson suggested that although many variables may only be measured subjectively, at least possible directions for research are being found.

Shackel [34], in an up-to-date (1970) consolidation of ergonomic research, emphasized the need for improvement in ergonomic design of interface equipment. As a single example, he mentioned the 'legibility of characters' on many computer generated CRT displays as being 'noticeably poor'.

These singular examples of M-C studies of current importance are in no means representative of the whole or even a large part of the research required for a complete integrated solution to the evaluation problem. They do, however, imply some of the types of variables that must be analysed to determine their influence on the overall information process. It is important to realize that M-C systems are currently built and their designers do analyse 'some' of the variables of the system which they feel are critical. However, the design is often done 'in vacuo' with respect to knowledge about the sensitivities of all variables on the total information process. Of course, to carry out the latter task would involve a very complex evaluation technique.

2.4 Approaches Taken Towards Evaluation

There have been a number of evaluation techniques proposed by various researchers. Because of the complexity of most Man-computer systems, many studies, as previously pointed out, have been limited to analysing only certain aspects of the interaction process. In fact, it is most unlikely that a single researcher would have the knowledge from experience, at least at a sufficient level, to pursue the total design of an M-C system.

However, some studies have been carried out on M-C research which represent a conglomeration of knowledge assembled by many separate studies from various disciplines. Sackman [35], in his 'Computers, System Science, and Evolving Society', has done extensive work in classifying overall Man-computer systems and emphasizes that evaluation techniques should integrate the design of such systems with their testing. His

detailed description of the SAGE system offers a frame of reference from which he systematically referred to in describing functional evaluation techniques. An underlying idea throughout this work was that '... the computer system should optimize around the characteristic variables of real time human norms for effective system performance rather than try to fit the human into an alien pace that may ostensibly be more convenient from program and equipment considerations'. This suggests that an M-C system should be designed with a complete knowledge of the user's requirements, viewing the man, not the computer, as supplying the constraints about which to design.

Various researchers have carried out less ambitious studies on smaller areas within the overall M-C system. In what might be described as a pioneer effort in the 'quantification' of displays, Siegel and Mickle [36] pursued the problem which they described as:

'... the need ... for methods and techniques for comparatively evaluating various design concepts before prototype systems and subsystems are actually produced. While subjective standards and appraisals possess merit in the absence of more objective and reliable measurement and evaluation techniques, the provision of quantitative system effectiveness and measurement techniques which could be applied early in the design stage might do much to eliminate costly design errors and expensive retrofits'.

Their work on a 'Display Evaluation Index' consisted of three parts:

- (a) Surveying and isolating components or variables in the index and getting measures for these components.
- (b) Developing ways of weighting the components.
- (c) Verifying the index against criteria for representative systems.

The approach taken was a mathematical one and some critics have sug-

gested it to be too complex and that the assumptions underlying the Index are too vague with respect to concepts of system effectiveness. However, it is an approach to evaluation in the area of M-C studies.

Gold [14], in a dissertation, proposed a methodology for evaluating time-sharing computer systems in terms of their usefulness for 'present' (1967) and future customers. It was based on a study of characteristics and design of computer systems as well as on 'relevant' behavior theory and research. This researcher included five categories of variables in this methodology, namely those which are measures of:

- (a) cost of using the system.
- (b) performance produced through use of the computer system.
- (c) speed with which results could be produced.
- (d) amount of learning resulting from the use of the system.
- (e) attitudes of the users of computing systems.

Gold's work was obviously concentrated on the Man-environment. In attempting to analyse the usage of time-sharing computers, he suggested four categories of variables relevant to an understanding of this usage, namely those which:

- (a) are characteristic of a computer system under study.
- (b) concern the attitudes of the users of these systems.
- (c) are measures of user's behavior and usage.
- (d) are measures of performance of a Man-computer system.

The relationship among these categories is illustrated in Fig. 2-7.

In 'A Parametric Approach to the Evaluation of Military Information Systems', Grimberg [15] reported on an evaluation technique applicable to M-C systems by 'cost-effectiveness' analysis. In this

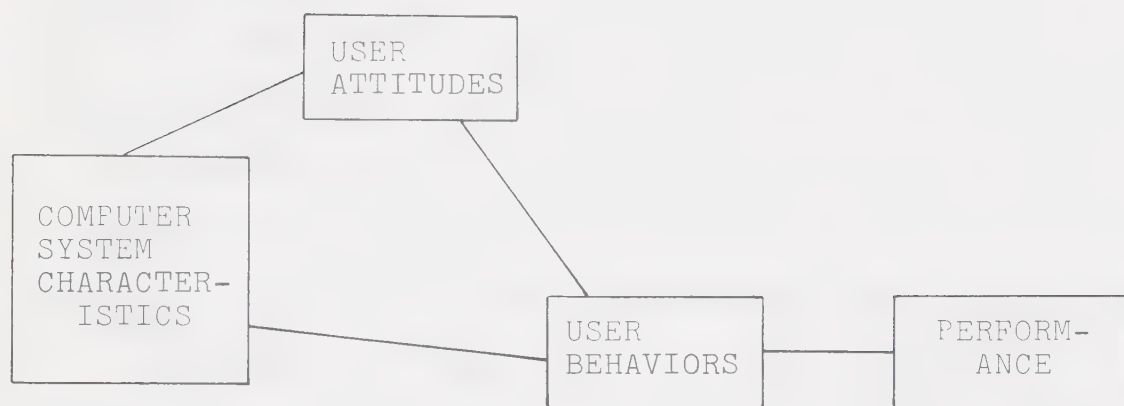


Fig. 2-7. Model of computer system usage (Gold)

approach, the effectiveness of such a system is measured by its impact on the effectiveness of the military operation using it. Five 'significant' parameters through which an information system affects the military operations are defined parametrically. These parameters were described as:

- (a) promptness
- (b) dependability
- (c) brevity
- (d) perspicuity
- (e) discriminance

This set was selected 'in order to make them unambiguous and measurable'.

Of the many disciplines which must be welded together to build Man-computer systems, probably the best defined, from an operational and functional point of view, is that involving the machinery of the Computer-environment. In this area, communication networks must be defined; an often massive and changing bank of data must be organized; terminal hardware and software must be chosen through which man and machine may interact; complex supervisory programs may have to be written to coordinate the many activities of M-C systems.

2.5 Structure of Techniques for Evaluation

The evaluation of any complex system can be a difficult, if not impossible, task when objective measures of its performance are required. The problem often appears more manageable when the system is divided into subsystems, each capable of being separately analysed.

However, with respect to M-C systems, various researchers have cautioned

that individual subsystem design is often done with too little regard for the eventual user of the integrated system.

Evans [12], in a paper describing a methodology for command information systems analysis, described '... a complex dilemma that exists between two communities: the user, who is operations oriented, and the designer who is technologically oriented'. One consequence of this situation is an 'inadequate conceptual understanding' of what an M-C system is and does, and why.

Evans cited examples caused by this inadequacy which suggested a basis for structuring an evaluation technique. Users of M-C systems, for example, tend to characterize and specify the type of Computer-environment needed from a base that is 'either too gross ('we need to plan faster') or too narrow ('we need a computer with a 16K core')'. On the other hand, designers, in spite of an intimate knowledge of existing technology and projected trends, are unable to create and put together 'relevant' design approaches. Often, according to Evans, their system designs incorporate 'unintentional' biases favouring one solution. Thus, detailed hardware/software specifications are not meaningfully related to the user in the Man-environment.

A difficulty is apparent in the structuring of an evaluation technique since it must work from two dissimilar bodies of information. The first is expressed as requirements from the user; the second is that of the better objectively defined performance variables of the Computer-environment. Hobbs and Braig [12] described the problem as '... on the one hand, there is often an explicit description of the functions to be executed by a number of black boxes with performance standards. On the

other hand, there is usually a vague, if any, description of the operational concepts and requirements for the overall system'.

It seemed apparent from the ideas expressed by many researchers, that any acceptable evaluation technique must be structured around the requirements of the system as directed by the user in the Man-environment. A technique must be capable of operating on user requirements, in whatever terms they may be measured, and mapping them into some 'optimal' Computer-environment configuration. Throughout this study, optimal refers to configurations which best facilitate user requirements in an economically feasible manner.

2.5.1 Need for a Methodology

A designer of subsystems associated with the Computer-environment has before him a bewildering array of hardware and software from which to choose. Further complicating the selection is a jungle of equipment options. This makes the number of equipment combinations available to a system designer large indeed. It is apparent, then, that a systematic selection procedure is desirable which can quantitatively and qualitatively aid a designer in choosing Computer-environment configurations. Improper choices can have serious effects on the overall 'worthiness' of an M-C system for certain applications.

The need for an evaluation methodology was illustrated by Foley [13] in his evaluation of small computers and display controls. The subsystem referred to was the remote computer-display control shown in relation to an entire display system in Fig. 2-8.

Foley describes the unwieldly size of a data base necessary to represent the corresponding performance variables of all possible equip-

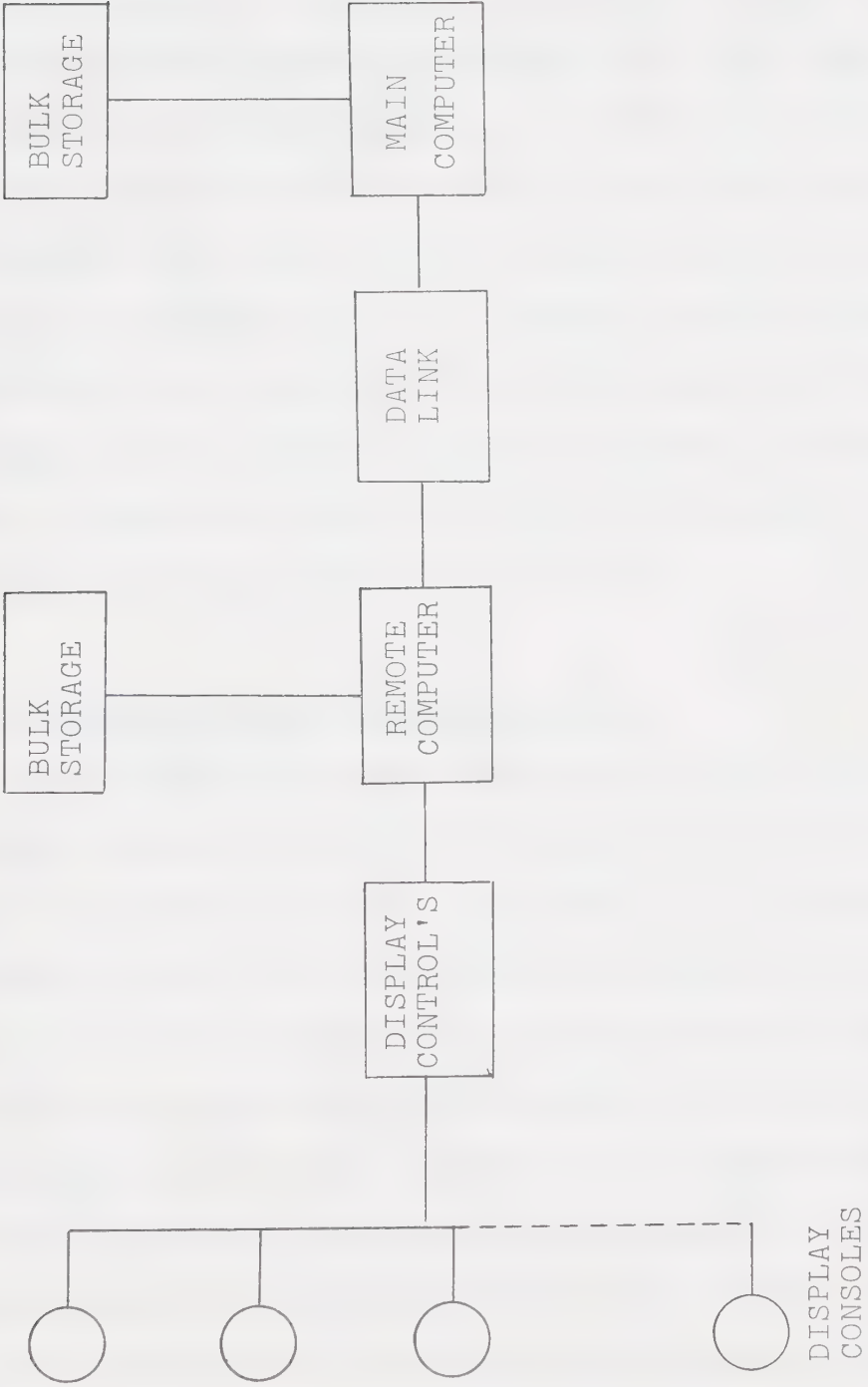


Fig. 2-8. General display system (Foley)

ment configurations and assorted options. There are currently several small computers which could be used as remote computers in a display system. They are characterized by a price range of from \$6,000 (or less) to around \$40,000, word sizes of from 8 to 18 bits, and are quite weak in performing mathematical (as opposed to logical) operations.

This illustration points out only one small area in an M-C system which requires evaluation. Taking all possible combinations of computers, display controls and options together amounts to thousands of possibilities. The magnitude of the problem expressed in this small area does, however, imply the much greater problem of evaluating total systems. Consequently, there is a need to develop good methodologies to aid in the design of optimal configurations.

2.5.2 Design and Validation of Methodologies

An acceptable evaluation technique must provide some sort of worthiness measure which represents a value of the system's overall usefulness in carrying out a user's application. Since the worthiness is measured in terms of 'how well' the user's requirements are met, research must be further exploited in the Man-environment in order to classify, label and develop measuring techniques which identify the elements of human input requirements. A methodology for evaluation must then work from these requirements and algorithmically translate them into characteristics of hardware and software configurations needed to satisfy the man.

It is desirable that an evaluation technique be as simple as possible as well as acceptably accurate. The technique used should be sensitive to all hardware and software features which are potentially avail-

able for implementation in the Computer-environment.

Once an evaluation technique has been decided upon, it must be validated to ensure its usefulness. Conceivably, an algorithmic procedure can be applied which works on a data base representing all combinations of equipments, and from a set of user requirements. A solution from this procedure should represent an optimal configuration.

However, determining that the actual selection(s) are reasonable is dependent upon examining them; determining that they are really correct requires that the applications be implemented with all combinations from the data base and the resulting worthiness of each measured by the user. It is simply the impossibility of this last task which motivated this research.

2.6 Subdividing the Computer-environment

In this study, little effort was directed towards describing the many types and operating modes of equipments found in real-time data processing systems. There is currently a large volume of reference material available which describes both the hardware and software functions of these equipments. However, since it appeared practical to subdivide the Computer-environment, some functional description appeared necessary in order to justify the particular divisional scheme selected. Such is given in both this section and in Section 3.2.

The research undertaken suggested various equipment groups capable of being considered as segregated subsystems. The choice used in the proposed methodology was based on those equipments falling within specific areas of M-C study currently being concentrated upon by various

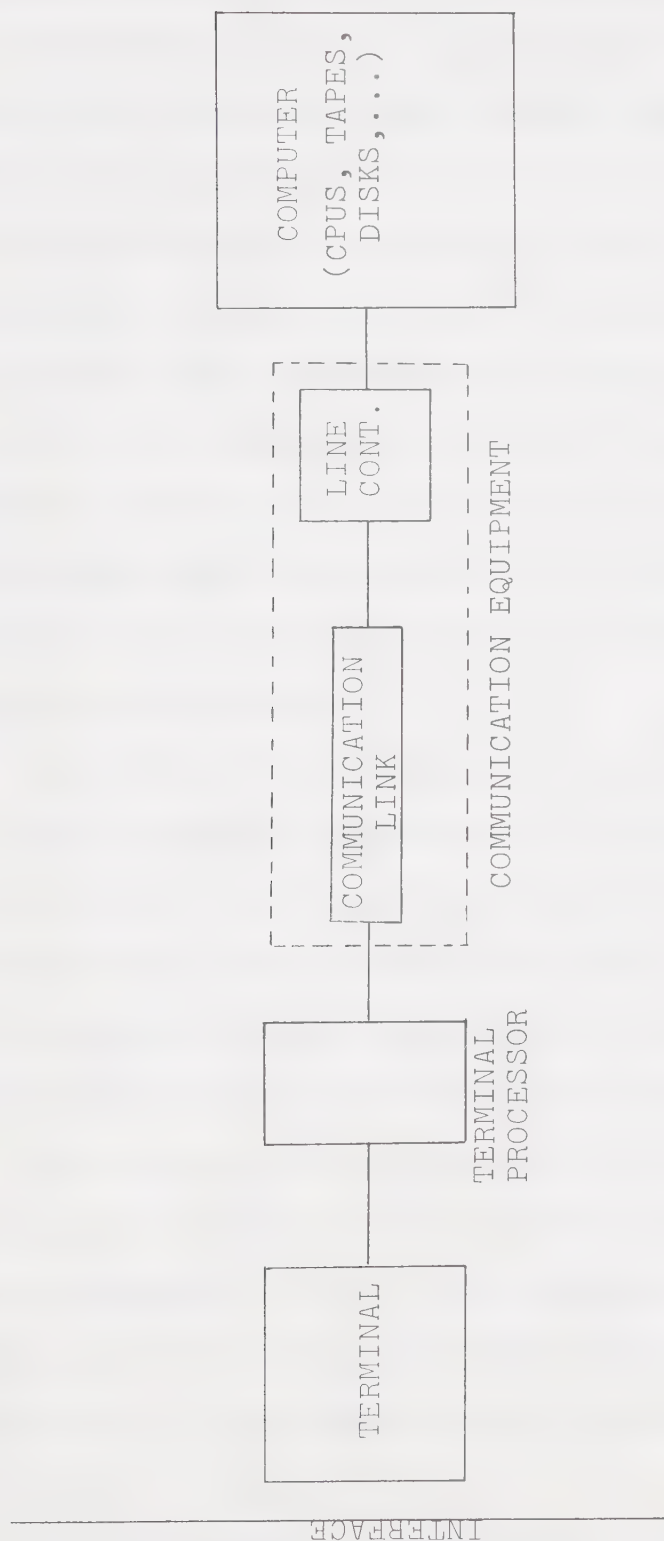


Fig. 2-9. Principle equipment groups of Computer-environment

researchers. In this way, it was felt possible to group relevant research findings according to the subsystems being analysed.

The research undertaken indicated that the principle equipment groups in this environment could be identified as those illustrated in Fig. 2-9. Among these groups, only the terminal equipments are considered as directly accessible by a user at the Interface. However, the terminal processor, which could be considered as an auxiliary terminal, is also often considered as Interface equipment.

Of the five equipment blocks shown in Fig. 2-9, many researchers regard the terminal to be the 'bottleneck' in M-C systems. That is to say, the terminal is considered as the largest link in the chain capable of restricting the flow of useful information between man and computer, and vice versa.

The communication equipment, shown in Fig. 2-9 as enclosed by dotted lines, was generally considered by researchers to be 'non-critical' in M-C systems. This is because research in such areas as communication and transmission control is considered relatively stable compared to the more 'undefined' elements of the chain. Consequently, little mention is made throughout this report about these equipments.

The remaining equipment groups, the terminal and computer processors, are considered by most researchers as compatible in terms of their developement. In fact, their equipments often function identically, such as in the case where the terminal processor is actually a small computer. In the limiting sense, however, the terminal processor may merely be a message buffer.

Much of the research undertaken for this study was concerned with

evaluating terminal equipments. This is not to suggest, however, that bottlenecks do not, or cannot, appear via other equipments in the Computer-environment. Instead, much less information is currently accessible regarding the Interface equipments than the more studied equipments, such as the computer.

Subsequent to the background research, it appeared reasonable to concentrate an evaluation technique on only those equipments contained in the terminal and computer groups. Moreover, because of the currently proliferating number of terminal devices on the market, the former group was separated into input and output equipments. This was prompted by Shackel [34] in his recent ergonomic research findings. Consequently, the evaluation technique described in Chapter III considered three subsystems: Input, Computer and Output.

2.7 Selection of Design Parameters for Computer-environment

It was pointed out in previous sections that a large amount of the researched literature discussed philosophy, problems and approaches taken in designing real time systems. Topics varied from detailed, explicit statements of user needs to speculation about the use of systems.

Most discussion was directed towards the overall 'usefulness' of M-C systems. Usefulness, in this study, was regarded as with respect to a user's characteristics and the application being carried out. Clearly, how a user 'sees' a system depends on his own characteristics (proficiency, habits, etc.) as well as on the job being done at a terminal. Generally, a user will describe usefulness as a measure of 'how well the Computer-environment helps me perform my assignment'.

For practical purposes, usefulness must not be considered alone, but with respect to the price the man is paying for it. According to Carbonell [3], the cost of an M-C system is composed of two factors: those associated with both environments. The Computer-environment presents such items as 'hookup flat charges, console elapsed time, CPU usage, etc.'. The Man-environment cost is seen by two factors. One is the user's cost as a scientist, programmer, etc. derived from salary. A more important one is that cost of postponing alternative activities.

Such terms as flexibility, speed, cost, accuracy, efficiency, discriminability, reliability, etc. are often used to qualitatively describe a conglomeration of characteristics. But seldom is there any apparent uniformity or standardization among researchers of what characteristics these terms actually describe. Clearly, for every user/designer, a mental idea does exist for the qualitative terms he uses, because without them an M-C system could not be designed with a specific purpose in mind. The qualitative terms appear to fall within two categories:

- (a) those used to describe user requirements of a system.
- (b) those describing the performance resulting from a system.

A set of descriptive terms used to 'judge' an M-C system's worthiness were called design parameters in this study. A major purpose of this research was to select an exclusive set of such parameters. This was done by reviewing various applications and their corresponding user requirements with the intent of selecting terms within which all equipment characteristics might fall.

Chapter III describes the selection of a set of design parameters. The methodology is based upon their definitions which in turn are user/designer selected.

CHAPTER III

THE METHODOLOGY

3.1 Overview of the Methodology

The research reported here is oriented towards describing a methodology to evaluate selective subsystems of the Computer-environment, as identified in Section 2.5. It is necessary, however, to realize that any total M-C evaluation must analyse the complete closed information loop. This implies that both the Man and the Computer environments would need to be evaluated in a total case.

The evaluation technique described in this chapter points out a method of obtaining Worthiness indices corresponding to subsystem configurations in the Computer-environment. Such measures of usefulness to a user are shown to be analytically derivable from the Design Parameters. These, in turn, are described through their Characteristic Equations. It is emphasized throughout that the usefulness of the technique proposed is largely limited by the accuracy with which a user/designer defines the Characteristic Equations for a subsystem.

An iterative process is outlined whereby hardware/software configurations may be tested, compared for relative optimality, reconfigured and retested until some best subsystem can be considered definable. By analysing Worthiness versus cost, a decision can be made of whether or not a best configuration can be considered within the set tested. Fig. 3-1 illustrates the principle operations involved in this process.

This chapter describes the principle concepts and definitions deemed necessary to project the methodology as being both conceivable and pract-

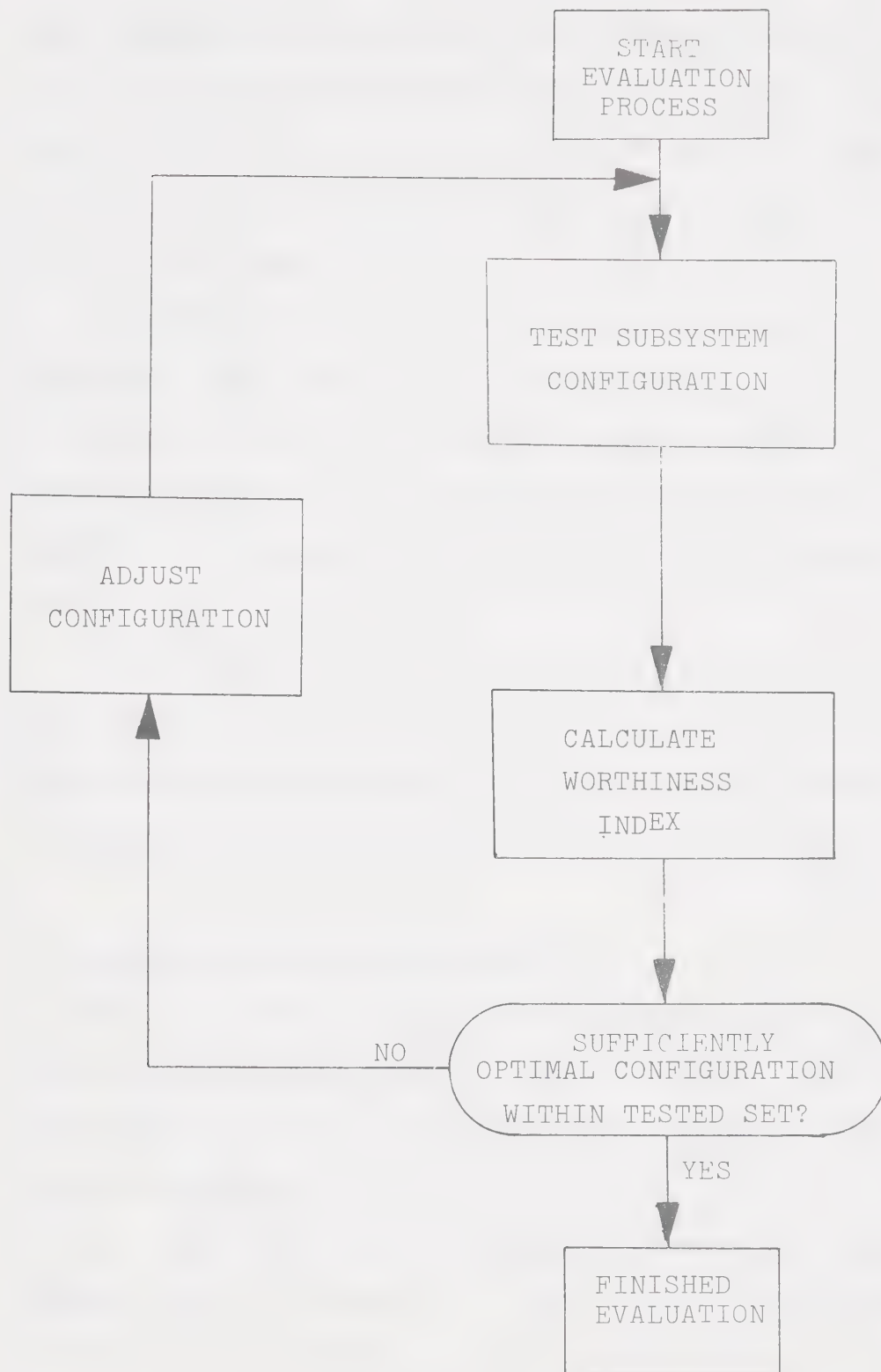


Fig. 3-1. Principle operations required by evaluation technique

ical. Section 3.2 describes the model of the Computer-environment used by outlining the basic functions of the three subsystems in general terms. Also, the idea of an External environment is introduced to allow for any additional loading seen by the Computer subsystem beyond the single on-line user.

For purposes of conveniently grouping all devices and their characteristics, the concept of a Requirements Space was introduced. Section 3.5 outlines how one may consider this space to be defined by three intersecting subspaces, each associated with a subsystem. This space is treated as a hypothetical data base from which a reconfiguration algorithm may operate upon.

Finally, the complete evaluation technique is described with respect to its building blocks. A hypothetical example is demonstrated and Worthiness indices are calculated. The chapter is concluded with suggestions of limitations of the overall capabilities of the methodology.

3.2 Model of the Computer-environment

Fig. 3-2a depicts the main hardware/software equipment groups considered in Man-computer studies. These three groups, namely, terminal, communication and computer equipments, are shown with relation to the External environment.

For reasons described in the previous chapter, the communications equipment was not considered as a usual bottleneck in the information flow process. Consequently, it is not further discussed throughout this study. Fig. 3-2b illustrates the exclusive equipment groups considered as vital in limiting the 'closeness' of interaction between Man and machine.

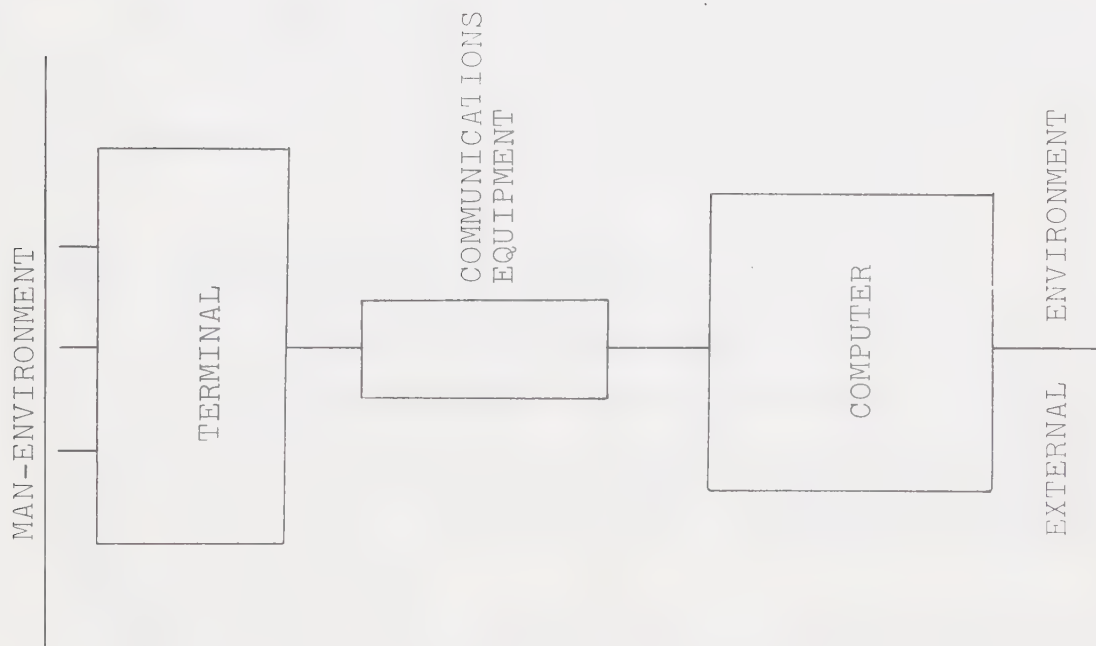


Fig. 3-2a. Main equipment groups of M-C system

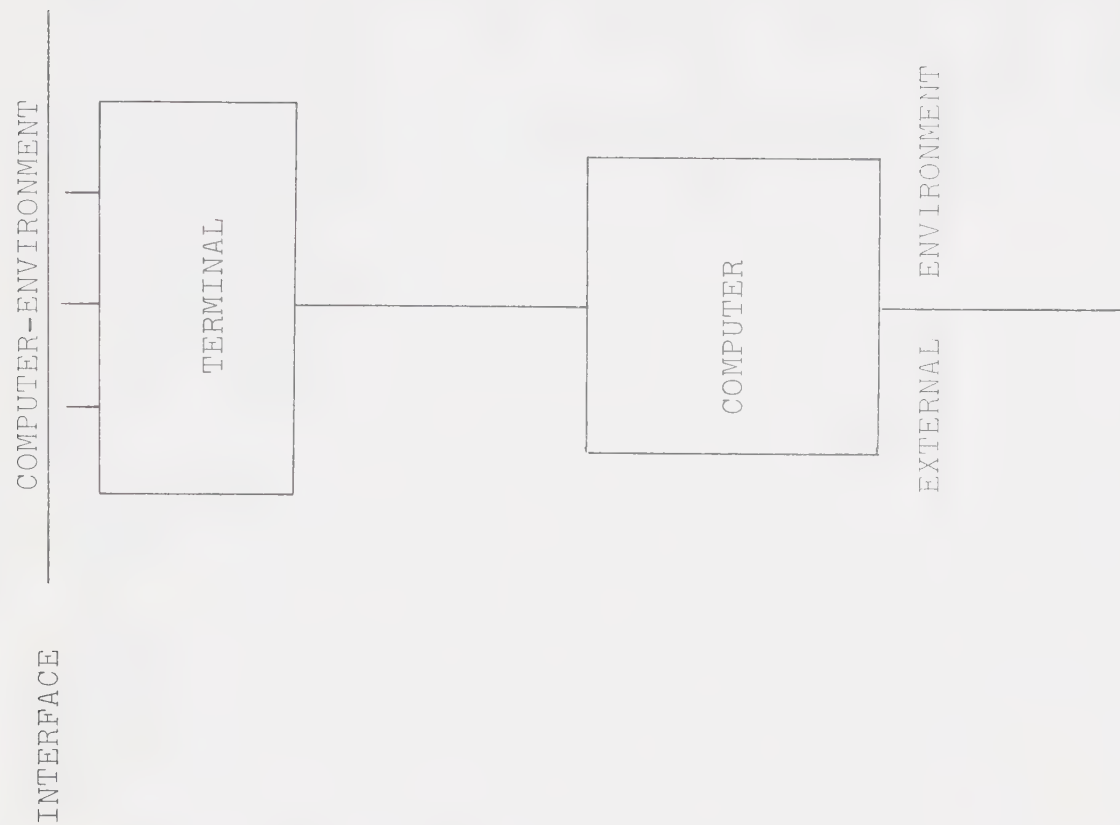


Fig. 3-2b. Equipment groups considered in this study

A further division of general terminal equipments was prompted by various ergonomic research findings in the Human Factors field. Specifically, it was apparant that often research references fell within either input or output type equipments of terminals. This is not always true since some literature refers to 'interduct' [34] equipments as being combinations of both.

Consequently, the subdivision of the Computer-environment was based on both operational and functional distinctions between groups of equipment of current interest in M-C studies. The subsystems considered as effectively linking this environment are modelled in Fig. 3-3, and shown in relation to the Man-environment. The solid connecting lines represent information flow channels, usually electrical signals on the computer side of the Interface. The segmented line represents an off-line control function and will be described in Section 3.2.1.

The function of each subsystem is described as well as the concept of an External environment. The level of detail used was that considered necessary to describe the steps leading to the methodology.

3.2.1 Function of the Input Subsystem

The function of the Input subsystem can be described as being of two essential modes: namely, on-line and off-line functions. The former mode was the only one considered of direct importance in this study. However, the off-line function is described to clarify the difference between the two.

The off-line mode of operation is that described by the segmented lines in Figs. 3-3 and 3-4. This represents a direct control function

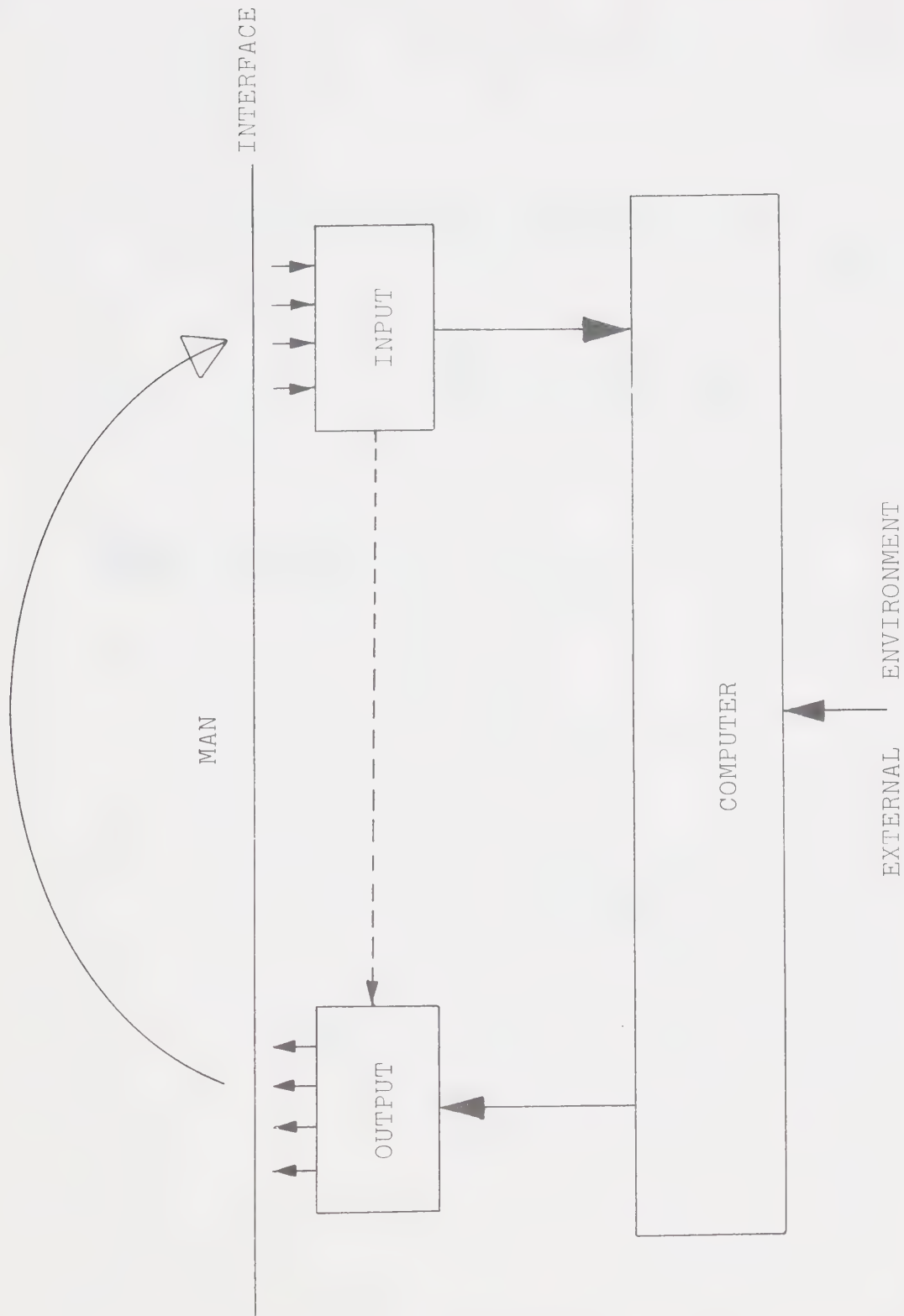


Fig. 3-3. Model of Computer-environment used in this study

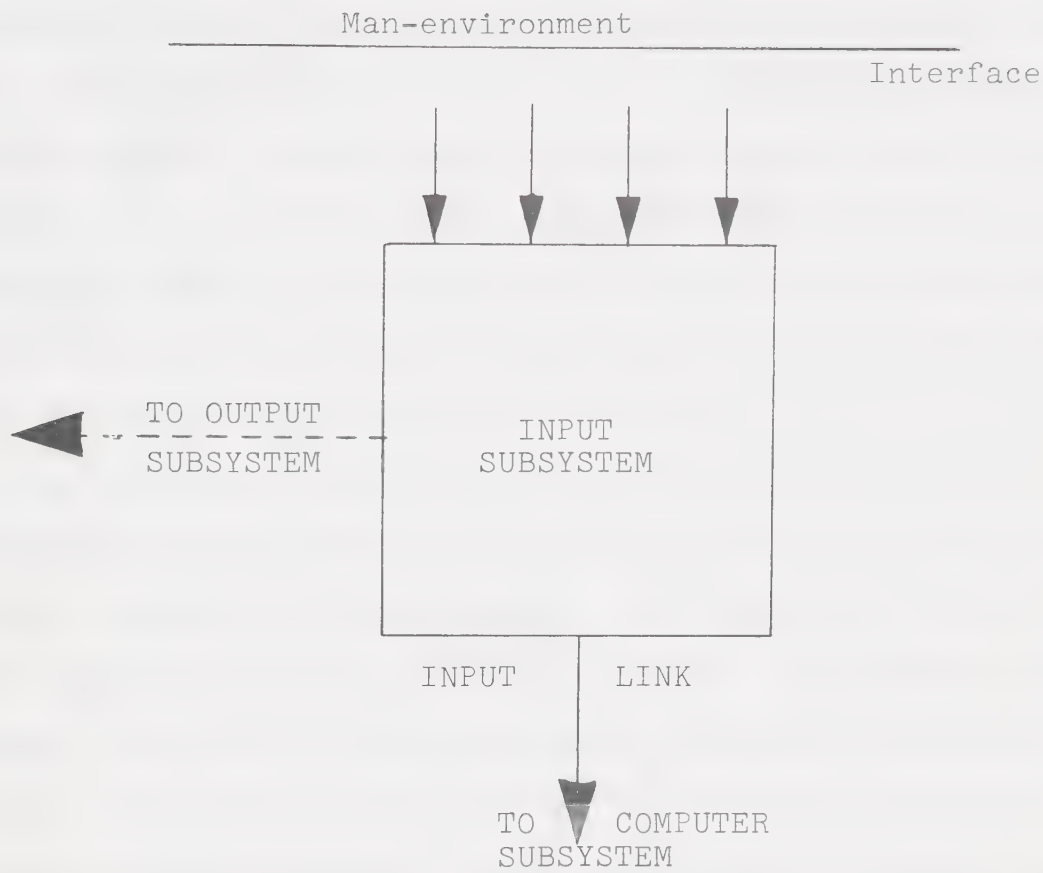


Fig. 3-4. The Input subsystem

to the Output subsystem. An example of a function employing this line might be that of an Input subsystem simultaneously producing a hard copy of an input transaction to the computer as input messages are formed. A typewriter keyboard input device, such as the IBM 2741, produces such a hard copy on the same paper medium as that used for output as directed from the computer. Another example of the off-line mode might be that whereby two or more teletypes communicate with each other directly, bypassing the computer. It is important to recognize the difference between user needs to input terminals requiring off-line functions compared to those needs directly related to the M-C loop.

The on-line mode referred to above involves information signals flowing through the Input link of the M-C loop (Fig. 3-4). In this mode, the only function of the Input subsystem is to transpose a variety of human originated electrical or mechanical signals, via transducers, into messages. The format of these messages is dictated by the communication equipments forming the Input link as well as the Computer subsystem which must decode the messages. The Input subsystem may contain a variety of support components such as remote computers, buffers, line control devices, etc. However, the function to this subsystem remains the same, irregardless of the sophistication of the equipments performing the function. For illustration purposes, it was felt useful to briefly describe two devices at the University of Alberta in their role as Input subsystems.

The GRID [20] system running under the General Purpose Supervisor offers a user the capability of entering various elements including light pen coordinates, function signals and alpha-numerical characters. This system, illustrated in Fig. 3-5, assembles a user generated sequence of

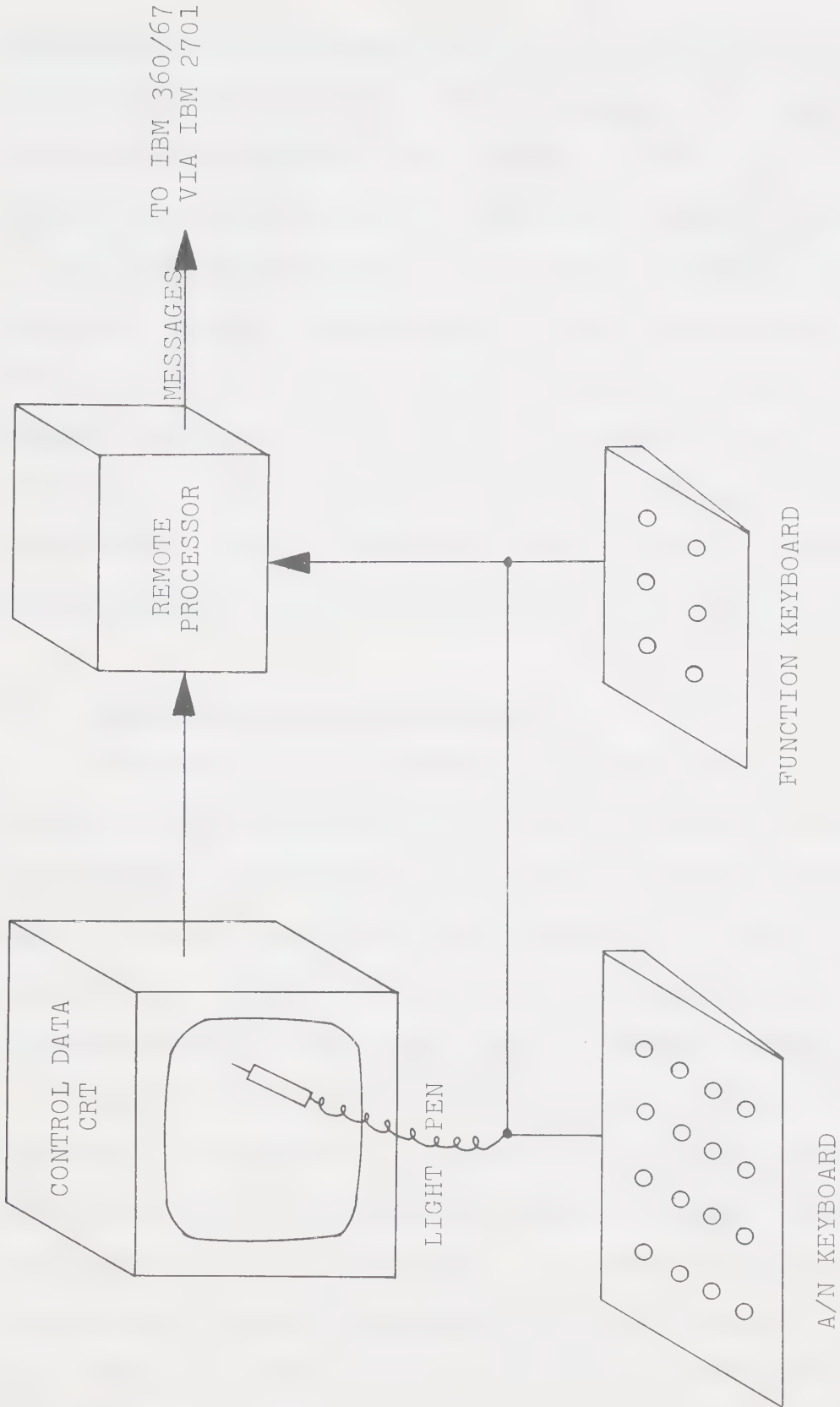


Fig. 3-5. GRID system as Input subsystem

input elements into a message consisting of two segments, namely a segment table and a message table. Only on command from the user does the processor send the message to the computer. In this role, the Input subsystem acts as an input buffer as well as a message formatter.

On a less sophisticated scale, the IBM 2741 typewriter terminal transmits a character code message each time a user depresses a key. Thus, there is no buffering at this input device (although partial buffering is essentiated by the IBM 2703 transmission control). Consequently, whether the messages are relatively large (as in GRID), or small, the function of the Input subsystem remains the same: to transmit user generated messages to the Computer subsystem in a recognizable format.

3.2.2 Function of the Computer Subsystem

The function of this subsystem, illustrated in Fig. 3-6, is to provide an equipment configuration of resources capable of translating input messages, performing operations implicitly requested by these messages and usually assembling response messages for a user in a format acceptable for translation by the Output subsystem. The input and output messages are considered as being digital bit strings throughout this study.

In general, the Computer subsystem can be viewed as a server to two work loads: one being the single requests from a user at the Input subsystem, the other being that created by all other inputs represented in Fig. 3-6 as the External Environment. Of course, the computer itself is only aware of a work load consisting of queues of requests for service from a variety of logical sources. The reasons for making this distinction are clarified in Section 3.2.4.

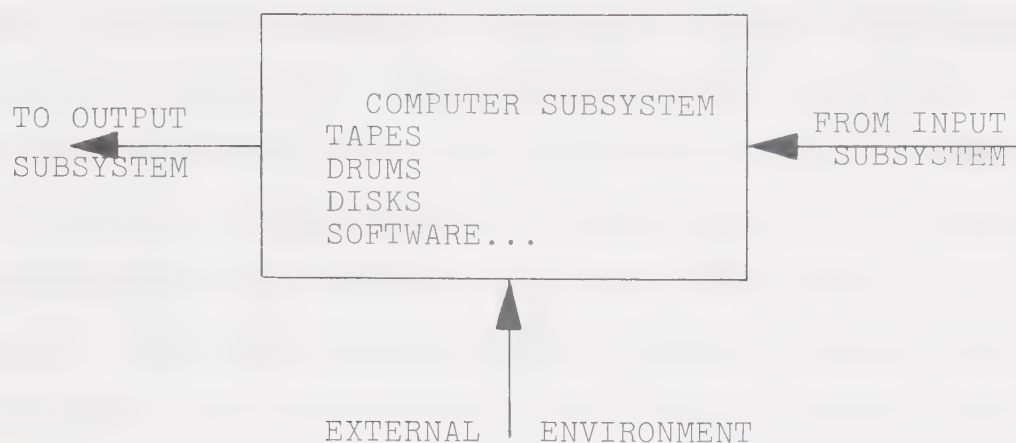


Fig. 3-6. Computer subsystem showing External Environment.

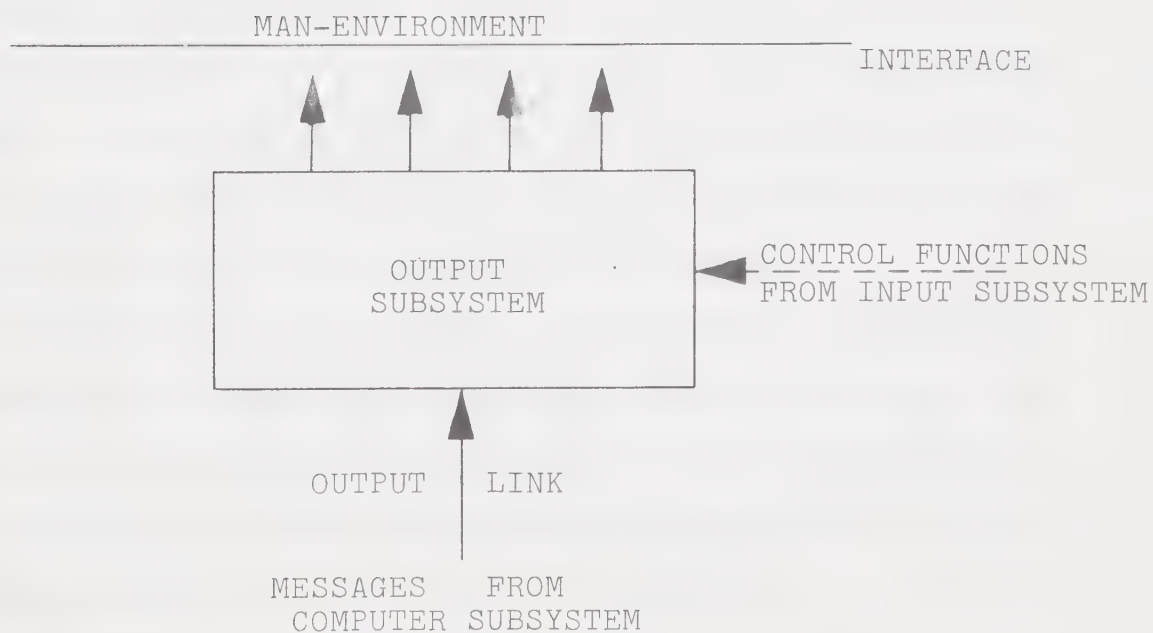


Fig. 3-7. The Output subsystem.

Generally speaking, the Computer subsystem is responsible for decoding input messages (via 'stored' hardware, software or flexware programs) into elementary requests for service. These requests implicitly describe a sequence of stored programs that must be executed on a variety of logical data sets. Usually, the exact sequence of programs run as a result of the input message is a function of the data bases each program operates on. On the basis of logic stored within the executed programs, the Computer subsystem may then assemble messages suitable for interpretation by the Output subsystem. Although various configurations of computer equipments may be assembled (such as different time-sharing systems or various sized machines), the functions are similar.

3.2.3 Function of the Output Subsystem

This subsystem, illustrated in Fig. 3-7, must interpret messages generated by the Computer subsystem and convert the implicitly defined commands into certain forms of signals receptive through human sensory channels. A secondary function, one of little direct importance in this study, is that of responding to control functions generated directly by the Input subsystem. This is illustrated in Fig. 3-7.

Functioning in the on-line mode, this subsystem must be receptive to messages transmitted via the Output link of the M-C close loop. The operation on these messages is complementary to that performed by the Input subsystem: an assembled message must be 'decoded' into an ensemble of analogue signals suitable to drive various devices.

For example, in the GRID system, operating under the General Purpose Supervisor, a message from the IBM 360/67 may implicitly contain

information on displays to be added, deleted or modified on the CRT screen. It is the function of the remote processor (refer to Fig. 3-5) portion of the subsystem to modify the display file according to the commands given via the message. On a lessor scale, the IBM 2741 operating as an Output subsystem must respond to a finite set of commands, one per message, instructing the terminal to carry out such singular functions as 'type a particular character', 'perform a carriage return (NL)' or 'logically identify yourself'.

3.2.4 The External Environment

The concept of an External environment was introduced to further clarify the approach taken in this study. Generally, concern was directed at one user 'looking at' a single Computer-environment. It was understood that this single user knows little about the computer beyond the fact that there must be some equipments available to support him.

However, although the methodology proposed considers a single Man-computer closed loop system, it seemed appropriate that consideration be given the more practical case where additional 'interference' may be subjected to the Computer subsystem. This interference was intended to represent all loading on the subsystem not directly attributable to the single user in the M-C loop. In some installations, this environment might be in the form of multiusers in a time-sharing system or signals entering a system from a real time process being monitored.

This terminology was used throughout this study, then, not because of any intent to analyse this environment, but because of the secondary effects on the Computer subsystem as seen by a user and caused by the

External environment. One such effect, of course, would be a fluxuating response time between interactions due to a varying work load on the subsystem. Unseen by a user, a 'slowdown' could be caused by conflicting requests for resource service by other inputs from the External environment.

In order to make the methodology as general as possible, the situation whereby a user in the M-C loop is in an external controlling role was considered. Such a situation is one where his decisions are based on events that happen outside the loop but are monitored by the Computer subsystem. Fig. 3-8 illustrates an arrangement commonly found throughout the SAGE system as well as one representing an industrial process control situation. Both may be conveniently dealt with, for purposes of this study, through the External Environment.

3.3 The Worthiness Index

Among the many terminologies used by researchers to describe configurations of equipments in the Computer-environment, two were prominent: capability and cost. Although researchers have concocted various meanings and suggested various ways one might measure these parameters, there appeared to be little uniformity among them.

Generally speaking, capability was referred to as the ability of a configuration of equipments to perform its 'intended' function. In this study, an intended function was Man oriented, i.e., the task a configuration should respond to was considered dependent on both a user's characteristics as well as his application. In this context, then, capability referred to a configuration's ability to satisfy the Man-environment.

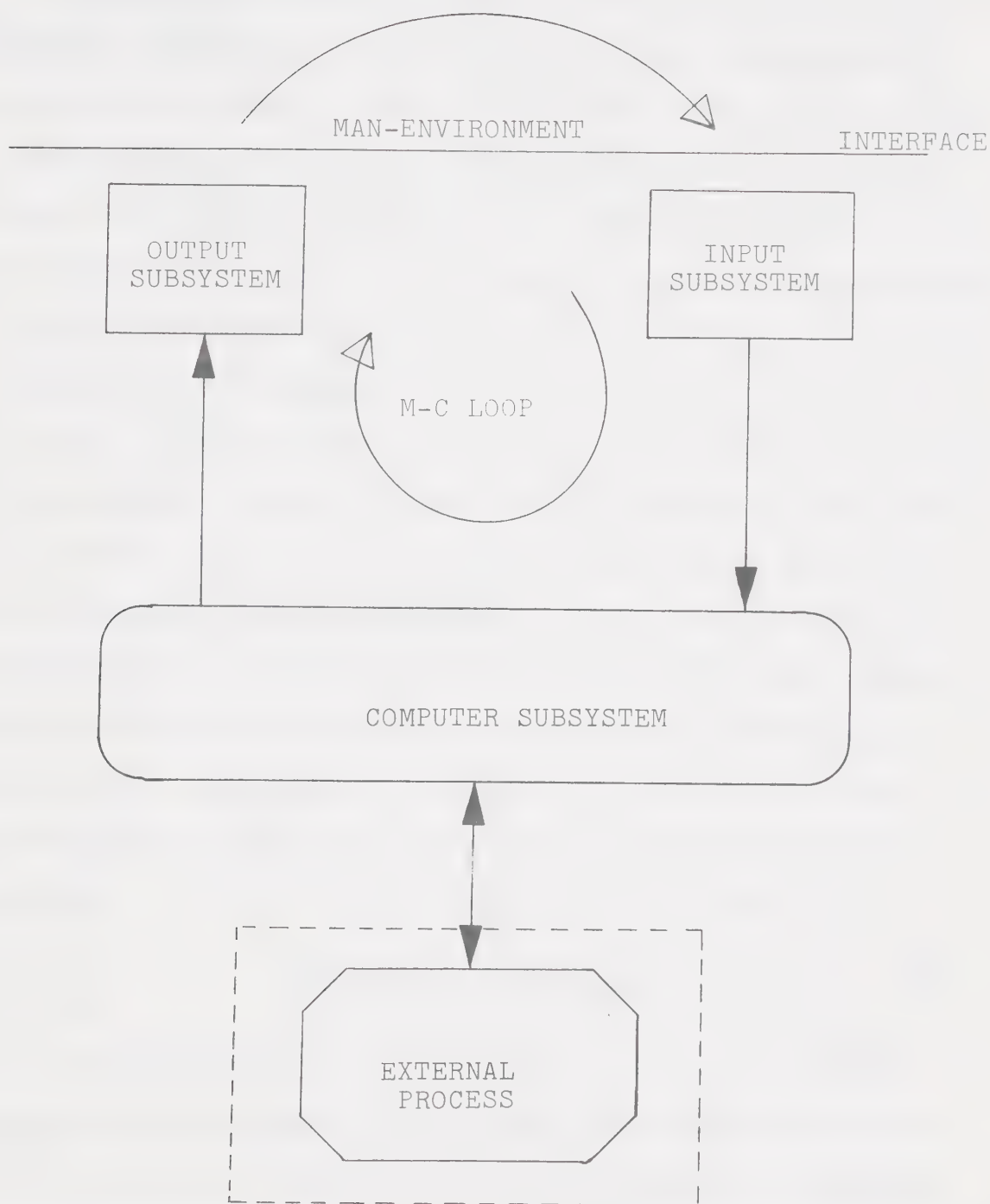


Fig. 3-8. A process control situation shown in relation to M-C closed loop.

Cost, the second parameter often referred to by researchers, was generally considered as a dollar measure of what a configuration of equipments was costing the user. In Chapter II, reference was made to a number of ways that researchers suggested measuring cost. In this study, the problem was fielded, as that of measuring capability was, by leaving the choice to the user/designer. The idea of not suggesting a specific definition was not considered a critical problem in the development of the methodology.

For practical purposes, it seemed conceivable that a user, if given the opportunity to employ his application on a variety of configurations of the Computer-environment, would find one which exhibited those characteristics which 'best' satisfied him. It also seemed reasonable to suppose that this configuration would not necessarily be one offering the most capability because likely that configuration would cost more than the user would 'measure' it to be worth. Consequently, a new terminology was introduced, namely Worthiness. In this study, Worthiness represents a function of both capability and cost, namely:

$$\text{Worthiness} = w(\text{capability, cost}) \quad (3.1)$$

The introduction of a new parameter to the already 'brimming' group of existing terminologies now used in M-C studies was not done merely for sake of originality. Rather, it was intended to utilize a terminology which was connotatively oriented towards the Man-environment. 'Cost-effectiveness', a term commonly found in research papers in a host of disciplines, was shunned for a variety of reasons. Foremost, this term has been and is used in many different contexts, and although some

are close in meaning to that desired in this study, it was considered unsatisfactory. Sackman [35], for example, used 'cost-effectiveness' in various ways including in his remark that:

'... determining the cost-effectiveness of computer systems is a highly complex affair and involves more than just considerations of hardware costs and operating efficiency'.

Worthiness was coined to refer to the effective usefulness of a configuration to a user. In most cases, a Worthiness versus Cost curve representing various configurations of the Computer-environment has the general shape illustrated in Fig. 3-9. The peak of the curve area should represent those configurations which best serve a user.

3.3.1 Concept of a Matched Subsystem

Further to the discussion of the Worthiness index, a 'matched' subsystem is described. The concept arose from the analog situation seen in the design of electronic systems. Two electrical networks, when connected together, must be impedance matched in order to ensure a maximum power transfer from one network to the other. This is to say that if the two networks shown in Fig. 3-10 were to be connected, the impedance Z_a (looking back into network A) should be as close as possible to impedance Z_b (looking into network B) to ensure an optimal power transfer between them. The networks would be matched.

To utilize this concept in describing subsystems of the Computer-environment, it was assumed that somehow a quantitative measure of user requirements to a subsystem could be made. Furthermore, since the capability of any subsystem was considered with respect to its requirements, it was assumed that a subsystem's capability could also be measured in the same units as requirements.

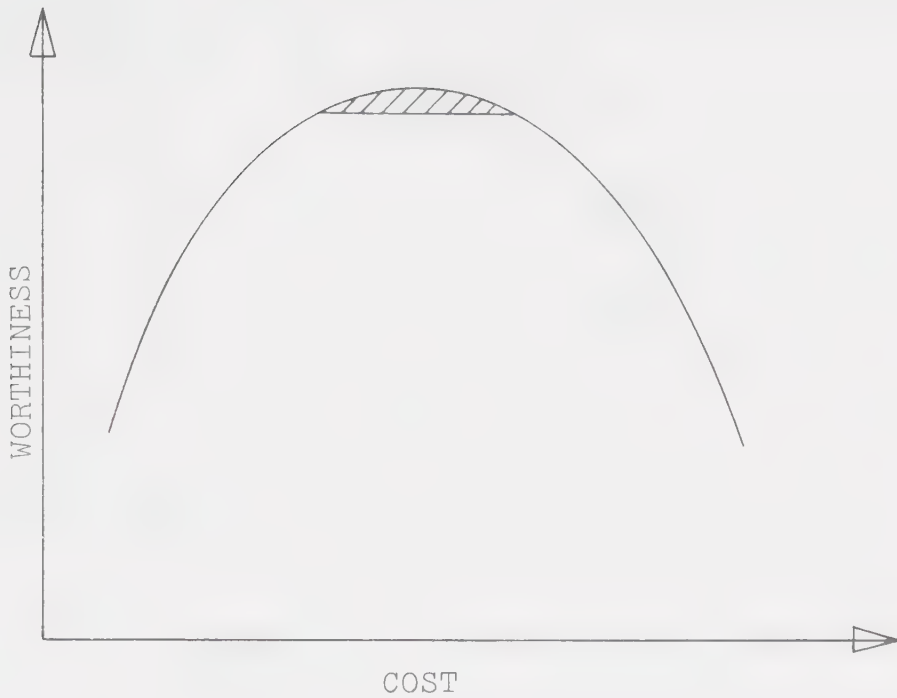


Fig. 3-9. Worthiness vs Cost for various hypothetical configurations of equipments. Cross hatched area represents 'best' configurations.

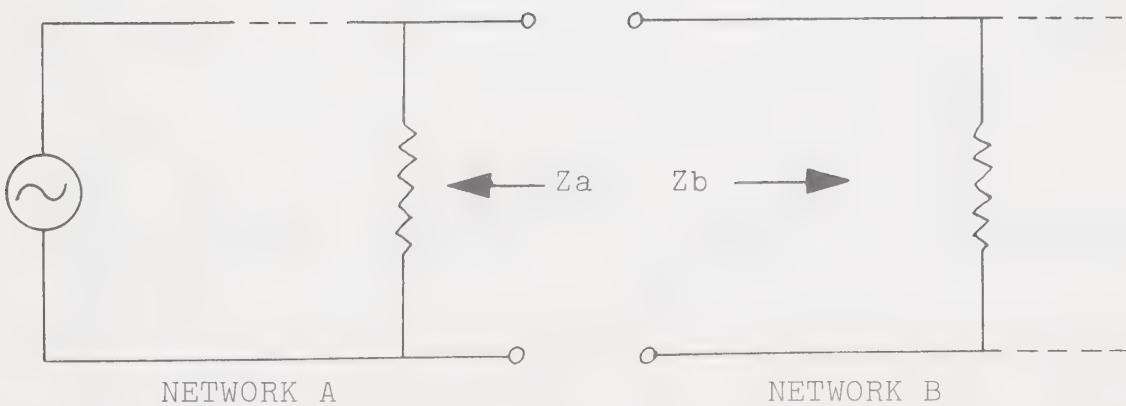


Fig. 3-10. The electrical impedance analogue for matched systems.

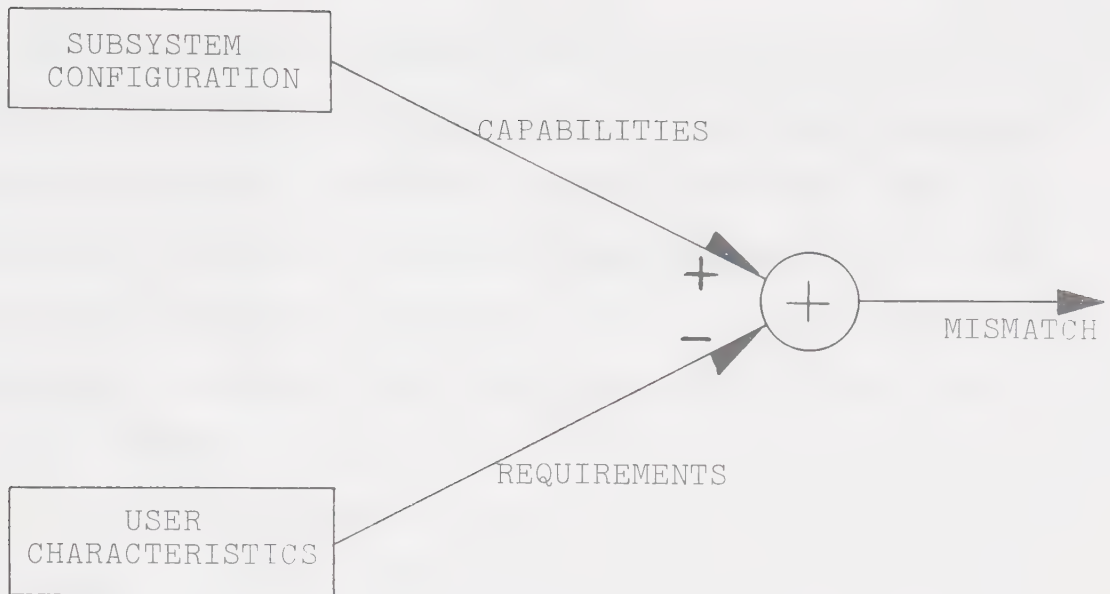


Fig. 3-11. Summing Junction Model to analyse capabilities vs requirements

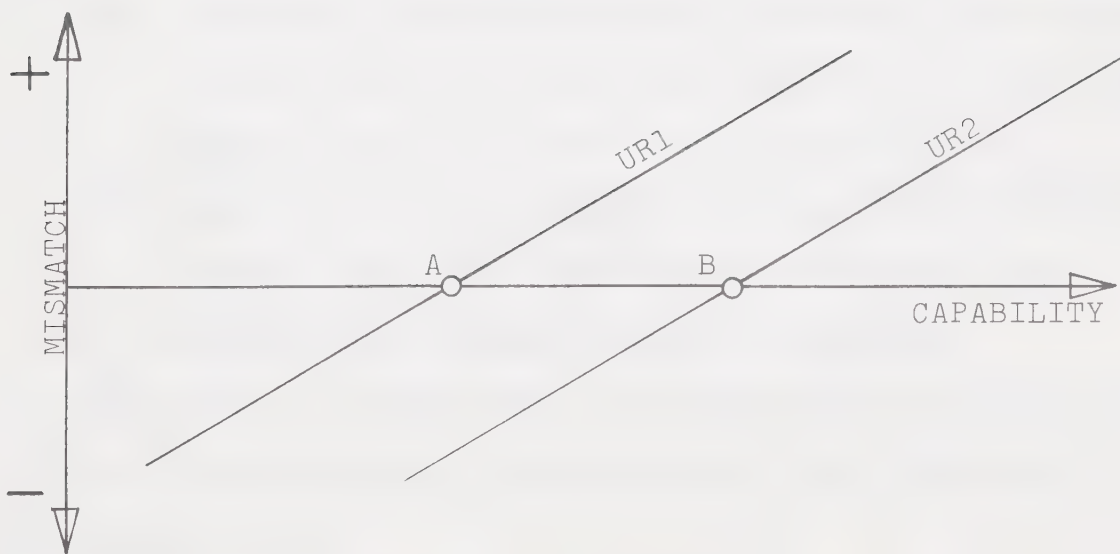


Fig. 3-12. Hypothetical curves resulting from varying configurations on two sets of requirements.

Fig. 3-11 depicts a summing junction, the inputs being both user requirements and capability of a subsystem. Requirements are shown negated with respect to capability for purposes of this illustration. Conceivably, as various configurations are analysed, for a particular and constant set of user requirements, various positive and negative differences could result from the junction. The difference signal is labelled 'MISMATCH' in Fig. 3-11.

If the junction were an algebraic summer, then the linear curves shown in Fig. 3-12 might represent the capabilities of various configurations as a function of the Mismatch outputs. Two curves are illustrated: UR1 and UR2, representing two sets of user requirements.

Conceivably, then, if both a subsystem's capability and a user's requirements could be measured in similar units, a subsystem could be 'tuned' to approach some matched (such as point A or B in Fig. 3-12) configuration. If a configuration was not matched to a user's requirements, one of two situations must exist:

- (a) $\text{Mismatch} < 0$: subsystem configuration is unable to meet demands of user described through his requirements. User may also be paying too much for an unsatisfactory configuration.
- (b) $\text{Mismatch} > 0$: a user is probably paying too much for what he requires. Most likely, the services he requires could be satisfied by a subsystem offering less capability.

Many researchers have implied that often with current M-C systems, the Input and Output subsystem configurations used might be described by situation (a) above.

Referring to Fig. 3-3, which describes the information flow through

the subsystems of the Computer-environment, it must be clearly understood that a mismatch in any of the subsystems will cause an overall impairment in the flow of useful information through the M-C closed loop. Furthermore, the subsystem causing the largest mismatch is essentially the limiting bottleneck of the environment. This is to say, it may be of little or no use to attempt 'better' tuning a subsystem which is not the limiting bottleneck before improving the worst part of the loop.

3.3.2 Response Time and Configuration of the Computer Subsystem

For purposes of this study, the response time of the Computer subsystem was viewed as the sum total of an alternating sequence of compute and wait periods. With reference to Fig. 3-6, an input message arrives to the subsystem via the communication equipment connected to the Input subsystem. In most systems, an interrupt signals the computer of a message pending for service.

Compute periods (C) represent those time segments dedicated towards processing the commands implicit in the input message. That is, CPU time given exclusively to the user's task; whether it be overhead for scheduling, servicing interrupts, paging, etc. for this user is unimportant for this discussion. Wait periods (W) include all time periods, between C-periods, where the Computer subsystem is servicing the External environment or waiting for an event to complete, such as I/O.

Then, from the Man-environment, a user might view the Computer subsystem as alternating between two 'states', C-period followed by W-period, etc. As far as a user is concerned, the response time for a particular request is simply the algebraic sum of these periods:

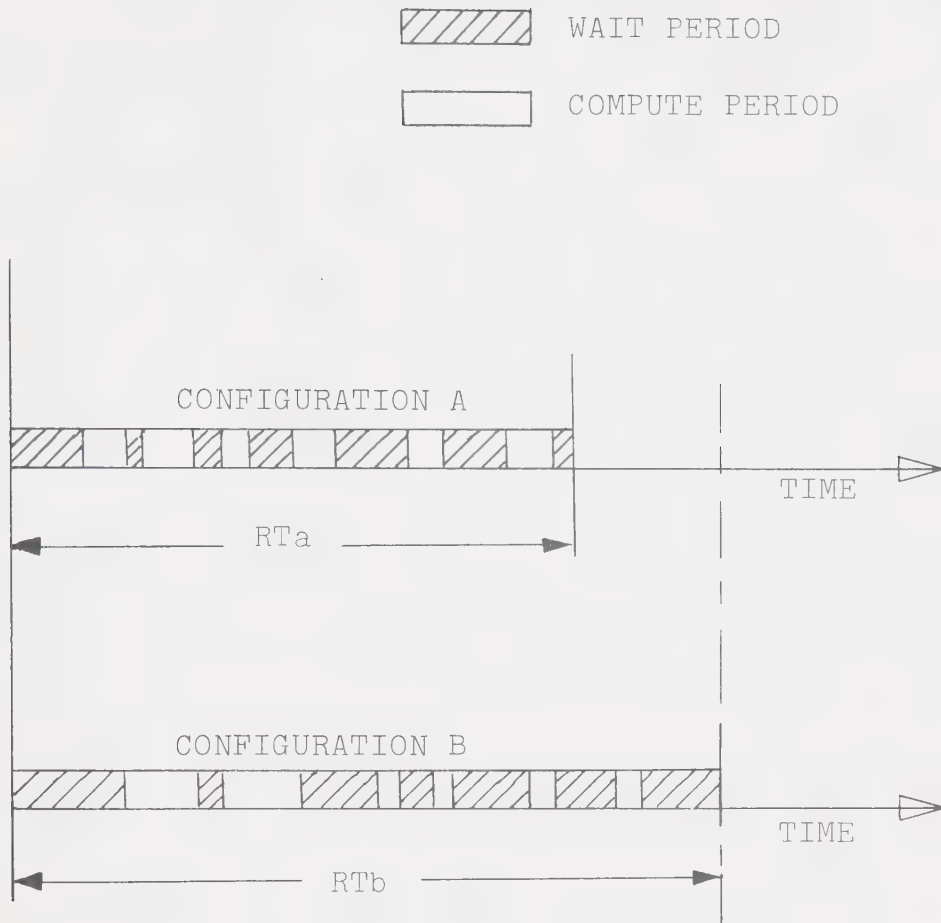


Fig. 3-13. C-W graph showing different response times for two different Computer subsystem configurations operating on same application.

$$\text{Response Time} = \sum_{T_{in}}^{T_{out}} \text{C-periods} + \sum_{T_{in}}^{T_{out}} \text{W-periods} \quad (3.2)$$

where T_{in} represents the message interrupt time on input and T_{out} is the time a reply message is finally transmitted from an output core buffer.

Various computer configurations, all capable of processing an input message, may provide various response times. Fig. 3-13 illustrates two C-W graphs representing two corresponding configurations.

The viewpoint taken to describe response times in this section was considered useful because no specification of the subsystem's equipment configuration is required in support of it. For example, whether a single or a dual processor is assembled (the latter being a multi-processing configuration) is unimportant as seen from the Man-environment. The C-W period concept retains its meaning, namely that the subsystem is currently either processing a user's message or not.

Of course, if the Worthiness indices corresponding to the two configurations referred to in Fig. 3-13 were examined, large differences might be apparent. In other words, the apparent improvement due to a 'better' response by configuration A may not be deemed as an overall improvement, especially if he finds he is paying more for configuration A than he feels the percentage improvement of:

$$\frac{RT_b - RT_a}{RT_b} \times 100 \text{ is worth.}$$

3.4 Concept of Requirements Space

The Computer-environment was considered to include a linked arrangement of three subsystems, namely Input, Output and Computer as illustrated in Fig. 3-3. Each subsystem was viewed as a self supporting con-

figuration of equipments arranged so as to provide the functional capabilities described in Section 3.2.

Throughout this report various references are made to devices and characteristics of these devices which could conceivably be arranged to form functional configurations. It is the intent of this section to introduce a conceptual model whereby devices and characteristics of them may be considered as being elements of a set implicitly representing all possible configurations of the Computer-environment. This set was called the Requirements Space (R - Space).

This space is considered to be composed of three partially overlapping subsets, namely the smaller requirement spaces associated with the three subsystems. Fig. 3-14 is intended to depict this arrangement and shows the three sets and their intersections as combinations of the labels 'I', 'C', and 'O' corresponding to devices/characteristics of the Input, Computer and Output subsystems respectively. The intersections are labelled as 'I - C', 'I - O', 'O - C' and 'I - O - C'.

The R - Space is considered to include only unique elements, that is, every element is defined in this space only once, whether it corresponds discretely to a device or a characteristic of a device. Because of the not uncommon possibility whereby certain devices and/or characteristics may be configured in more than a single subsystem, the intersections shown in Fig. 3-14 are conceivable. Examples of this are given.

Conceivably, the Input subsystem might be configured to include certain types of auxiliary storage input devices, such as tape drives, which are also common equipments in the Computer subsystem. Although such devices may be smaller units at a terminal, they are functionally similar to those found in the computer. An element conceivably part of the 'I - O -

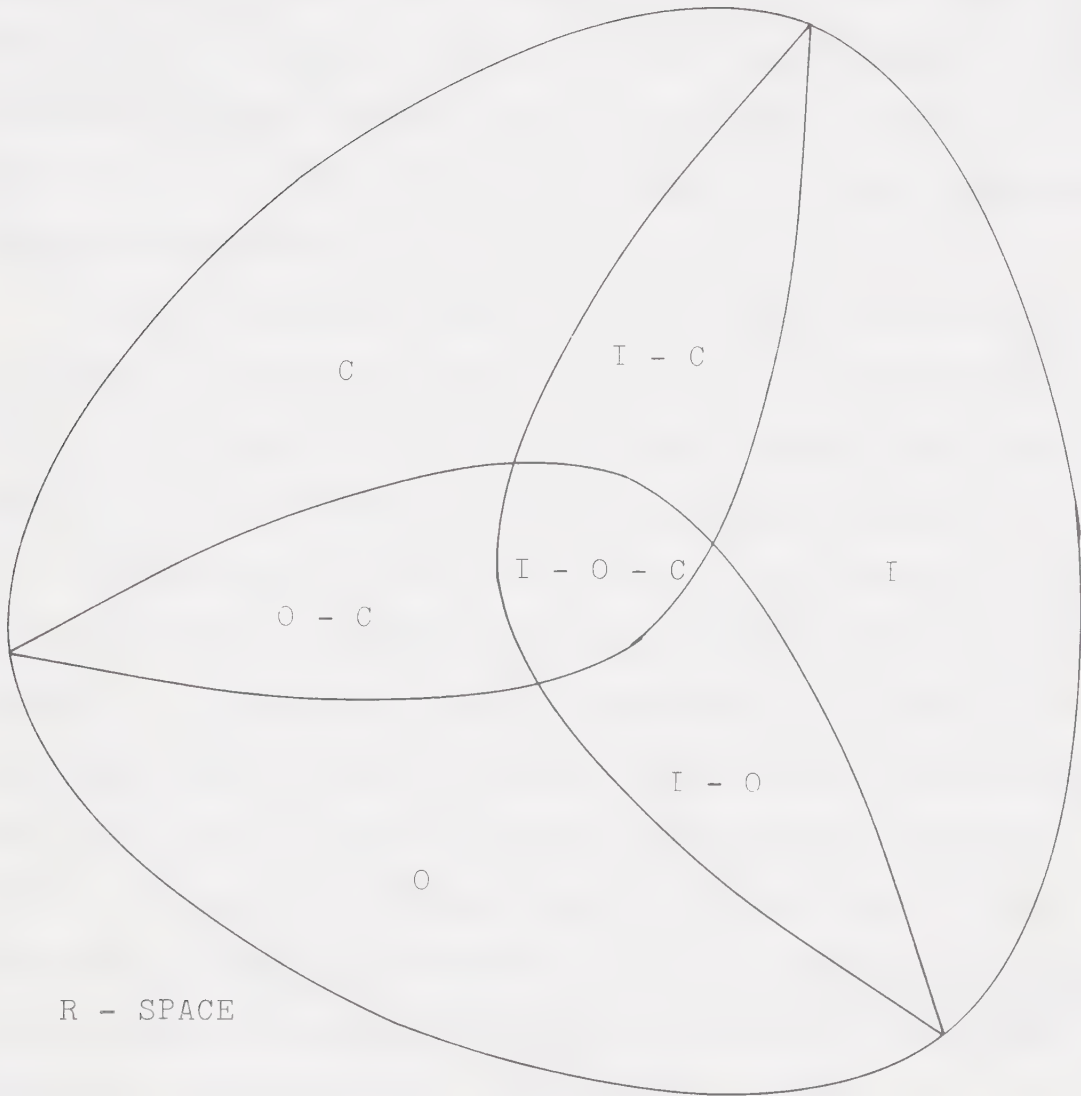


Fig. 3-14. R - Space consists of three subsets, plus their intersections, corresponding to subsystems of the Computer-environment.

C' intersection might be one corresponding to a core storage device. This element may be seen in the Input subsystem in the form of an input message buffer; in the Output subsystem as a display core (such as the GRID system); or as seen in most present day digital computers as CPU accessible storage. An Example of an element seen in only the 'I' sub-space would be a light pen device used to input coordinate information in the Input subsystem.

The above examples are those describing device type elements in the R - Space. Characteristic type elements associated with the devices also exist in this space. Examples of such might be magnetic tape speeds, track and density capabilities, light pen responsivities, etc.

When viewing the proposed evaluation technique as a whole, the value of introducing the R - Space concept becomes apparent. It is pointed out in Section 3.6 that the R - Space is considered as a data base of information containing all possible devices and characteristics of these devices upon which an evaluation algorithm must operate in a search for optimal configurations. Of course, not all combinations of elements from this space are meaningful, suggesting that any 'forbidden' combinations must be somehow detectable by the evaluation technique. For example, the brightness characteristic type element associated with a CRT screen has no (apparent) meaning when associated with a core device.

3.4.1 Requirements to a Subsystem

As viewed by a user/designer, a subsystem of the Computer-environment has two requirements, namely those referred to as:

- (a) design requirements, and
- (b) functional requirements.

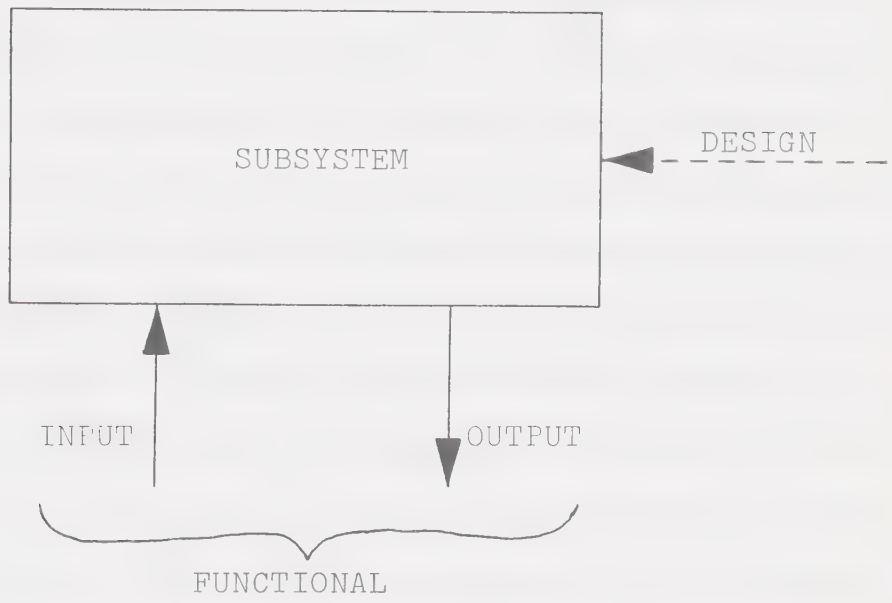


Fig. 3-15. Each subsystem of the Computer-environment has both functional and design requirements.

Both are illustrated in Fig. 3-15 by segmented and solid lines.

Functional requirements are those usually expressed in terms of electrical or mechanical input signals to and output signals from a subsystem. They refer to the function a subsystem must perform, but not the manner in which the function is to be performed. Also, functional requirements in no way imply any degree of 'satisfaction' a user may feel regarding the functional performance of a subsystem.

On the other hand, design requirements of a subsystem reflect the manner in which the functional requirements are handled by a subsystem. These requirements must, of course, be described by the user/designer: it is this complex task around which this study was focused. The design requirements are implicitly described through the Characteristic Equations, Price functions and Weights of Design Parameters (refer Section 3.5).

An example to further illustrate the distinction between the two requirements was felt worthwhile. Suppose a user desires that the Input subsystem be used to enter strings of alpha-numerical characters. This subsystem must respond to this functional requirement by exhibiting equipments capable of recognizing the characters entered, formatting the signals into messages and passing these messages to the communication equipment connecting the computer. Among the multitude of input devices capable of handling this task, a telephone dial and a keyboard device are two. However, a user would likely not be appethetic to the idea of 'dialing' in character strings at a terminal if, for no other reason, because of the resulting speed and accuracy limitations compared to a keyboard. These 'conditional' qualities defining how a user wants a particular function carried out are design requirement.

3.4.2 Functional Requirements at the Interface

From the research undertaken for this study, it was felt that it should be possible to describe a finite set of requirements to the equipments seen directly by a user at the Interface. This idea arose from studying various M-C applications and observing that although many researchers described 'their' applications as requiring a variety of inputs and outputs, there were apparent similarities among the requirements. It appeared that although every application required different sequences of input/output operations, these sequences were actually composed of a limited number of functional requirements.

The idea of selecting a finite set of basic functional requirements then seemed conceivable and for purposes of this study, warranted. Since various researchers have expressed the idea that it is desirable, from a user's point of view, to carry on a conversation with a computer in as 'natural' a way as possible, this suggested a criterion for selecting the elements of the set. A natural communication implies one normally possible between people. Therefore, a set was selected whose elements connotatively described the essential information in Man-man (M-M) communication. This set is not advanced as an exhaustive one, but one which was felt adequate for purposes of this study.

The set chosen represents the basic functional requirements to the Input and Output subsystems and is:

- (a) character strings
- (b) coordinates
- (c) lines
- (d) function interrupts

The elements of the set are described below and the parallel between M-M and M-C communication is illustrated with examples.

Character strings: these refer to any series of impulses to the Input or Output subsystems. Each impulse represents a discrete quantum of information. A common example of a character string is that produced by a keyboard device; the parallel example in the Output subsystem is the output of a typewriter. A less obvious example is that information transmitted between the man and the Computer-environment in the form of spoken words. The words are actually characters, grouped in various ways and separated by blank characters, transmitted via equipments capable of encoding (or decoding) them. Generally, these functional requirements are classified as character strings and characterized by their variable length.

Coordinates: these refer to the input and output of a number of magnitudes; the total number (called the dimension) is used to determine positional information such as points in space. In M-M communication, a single written digit would be such a type. In contrast to character strings, coordinate requirements imply equipments capable of transmitting information of a fixed length. As an illustration of equipment within this class, a light pen used at a CRT display defines an Input subsystem configuration capable of inputting two dimensional information. The pen may input virtually any number of coordinates, but the size of each is fixed by X and Y raster units. In output equipment, the transmission of coordinate information could be done through a display screen capable of flashing single numbers to a user (one dimensional, for example). Of course, a typewriter could also do the same function, but for purposes of

this classification scheme, coordinate information normally requires less sophisticated equipments for transmission than character strings.

Lines: these refer to the functional requirement of entering or receiving a locus of points (implicitly defined, of course, by coordinates) in a direct manner to or from the computer. In M-M communication, for example, any mental picture of shape depicts this type. The situation of entering a sequence of points via a CRT and light pen then does not describe this functional requirement. However, a device such as the Rand Tablet [5], a horizontal, electro-static operated drawing plane, would be classified in this area. An example of an output device for drawing lines would be a CRT screen or a Cal Comp type plotting board.

Function interrupts: these refer to the transmission of essentially binary states via equipments in the Input or Output subsystems. In M-M communication, for example, a sound impulse could illustrate such information. A single character input, for example, from a keyboard device is not considered within this class because of the variable capability between types. Generally, equipments necessary to transmit binary signals are much less sophisticated than those for other requirements. Binary inputs are those such as function keys or other unary switches. In the Output subsystem, a light or alarm bell are classified as function interrupts capable of signalling states to a user.

It is apparent from the above illustrations that no hard and fast rules are implied from which to classify all devices within the four functional requirements. Conceivably, however, the classes can be viewed as describing groups of equipments capable of meeting functional requirements

of a user at the Interface. These groups in turn, can be viewed as supplying various levels of sophistication in terms of capability and cost to the user. For example, in a process control situation, an operator may only need to enter coordinate information (in the form of control settings) and a limited number of function interrupts. It is unlikely that he requires a CRT equipped with a keyboard device for this application.

3.5 The Design Parameters

The methodology developed in this study proposes an evaluation technique whereby the three subsystems of the Computer-environment may be analysed with respect to a user. Each subsystem is evaluated on a similar basis, namely one oriented around the definitions of a unique set of terminologies.

Choosing the set of terminologies was not a trivial nor an easy task. From the large number already used by researchers from various disciplines involved with M-C studies, it was considered possible to select some subset which could be considered appropriate. The criteria used for the selection was as follows:

- (a) Terms used should be those frequently referred to by other researchers. There appeared to be no advantage to selecting or creating new words to describe characteristics which were already used even though there were a variety of definitions.
- (b) Terms selected should be as uniquely distinguishable from one another as possible. That is to say, the characteristics that most researchers tend to associate with them should fall within

singular terminologies where possible. Of course, with some equipment characteristics, there are interrelations and trade-off situations between them, preventing a perfectly independent set.

The situation described in (b) above is diagrammed in Fig. 3-16. In this hypothetical case, the cross hatched areas are intended to represent those characteristics 'influential' in more than one terminology. Of course, some may be influential factors in more than one space, but the graphical concept depicting such combinations becomes difficult to illustrate. For example, the brightness of a CRT screen could be a relevant characteristic (depending on the user and his application) and may affect the operators reflex speed at a terminal which may also affect the 'comfortableness' of the operator as far as eye strain is concerned. If speed and comfortableness were chosen, then the characteristic of CRT brightness might fall within both terminologies.

A set of terminologies, called Design Parameters (DP's) in this study, was selected then with the intent that all other terminologies referred to by researchers of M-C systems might be inclusive. They are:

- (a) Accuracy
- (b) Reliability
- (c) Response
- (d) Facility
- (e) Flexibility
- (f) Cost

A key point of this research is that a user/designer may design an M-C system by analysing only these parameters and none others. It is

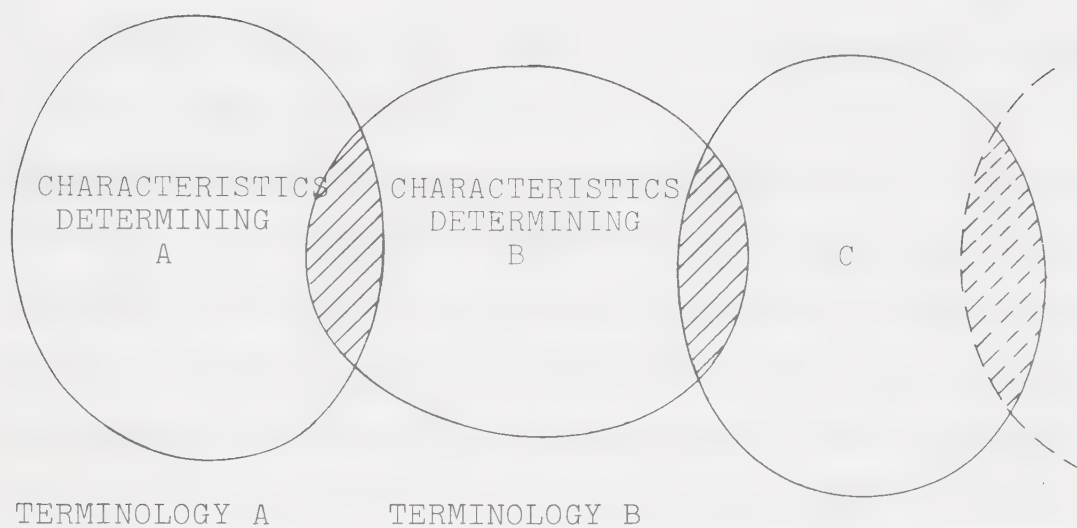


Fig. 3-16. Computer-environment characteristics should fall within a set of uniquely distinguishable terms.

not a trivial task to define these DP's since their definitions hinge on both user characteristics and their applications. They are considered measureable for each subsystem by way of Characteristic Equations which establish relationships between devices, and their characteristics, found in the R - Space.

General definitions of the DP's are given along with examples to help clarify their meanings. Exhaustive definitions were not possible within the bounds of this research.

Accuracy: This DP is described by those characteristics of equipment affecting the closeness with which the output of a subsystem approaches the user's desired or expected output. For example, a user may intend to enter a character string via a telephone dial: the resulting accuracy is likely severely limited by the characteristics of this input device compared to a keyboard. The intended value of real numbers obtainable within the Computer subsystem may be restricted by the floating point word size. Or an Output subsystem employing a pen tracking device may not be capable of tracing the accuracy of its input command signals.

Reliability: This DP is described by equipment characteristics which affect a subsystem's capability as a function of time. If the capability is constant, or nearly so, over a time period, the reliability as seen by a user may be considered high. Equipment that frequently malfunctions or breaks down might be considered as the source of characteristics affecting this DP. However, a user/designer may not be interested in an equipment's break down rate between certain hours, i.e., the definition is user oriented.

Response: This DP is generally regarded as concerning those characteristics affecting the speed with which information can be processed through a subsystem. For example, a character string could be entered at the Input subsystem via a telephone dial, a keyboard or from a remote paper tape device - the speed of input increases in this same order. Response time is used to describe the Computer subsystem's thoughttime for a message. The Output subsystem might have characteristics determining the speed that a response message can be made available for reception by human sensory channels. Actually, then, each subsystem has a 'through' time, however, the response characteristics discussed most often in M-C studies are those associated with the Computer subsystem.

Facility: Generally, this DP is determined by those characteristics affecting the ease of operation experienced by an operator at a terminal. It is not considered directly with the Computer subsystem. Such characteristics as key touch weights, CRT brightness levels, noise of terminals, etc. are considered within this complex parameter. Determination of the Characteristic Equation defining Facility is largely dependent on research in the Human Factors field.

Flexibility: This DP is often defined by characteristics determining the capability of a subsystem to handle additional requirements to it beyond those considered as basic user needs. For example, an Input subsystem incorporating a keyboard device with a variable character set might affect a user/designer's concept of this DP. Also, it is sometimes associated with the quantity and variability of equipments in the Computer subsystem.

Cost: In the Computer-environment, often the dollar worth of equipments, either through rentals, leases or outright sales, are attributed to this

DP. In this study, Cost includes all charges not associated with the Man-environment, i.e., hardware and software charges along with their operating costs.

A subsystem designer has some control over the magnitudes of the design parameters resulting from analysing various equipment configurations - control which is limited in several respects. First of all, as has been inferred, the DP's are not perfectly independent. For example, a designer may have to pay (tradeoff) for an increase in reliability by a decrease in the speed of a subsystem. There are regions which have no practical interest (for example, no capability at no cost). The design of a particular subsystem might be compared with a multi-variable problem in which a search for an optimum within certain boundaries or constraints is made. However, an optimum in one respect is seldom one in another, and necessary compromises must be found.

3.5.1 Relative Importances Between DP's

Section 3.5 described the conceivability of measuring absolute values for the six Design Parameters of a subsystem. In order that some sort of relative importance might be established between the DP's, the concept of Weights was introduced. For each subsystem being evaluated, a user/designer must supply a set of weights corresponding to various DP's being considered in that subsystem.

Consequently, with the introduction of Weights, the methodology operates upon effective DP magnitudes. Such values are the algebraic product pairs of the DP's with their respective weights.

For purposes of evaluating subsystems in this study, the Weights

were considered to be constant scalar magnitudes within the following boundary:

$$0 \leq \text{Weight of DP} \leq 1 \quad (3.3)$$

The lower limit in this equation represents the trivial case where a user/designer considers the characteristics capable of defining a particular DP to be unimportant or not meaningful for a subsystem. The upper extreme represents a situation where a DP is considered as 'most' important in respect to the other design parameters.

3.5.2 Price Functions of DP's

The researcher Gschwind [16] prompted the inclusion of Price functions into the methodology. In his choice of design parameters, he discussed some of the interdependencies between them and illustrated hypothetical curves showing pairs of parameters. For example, Gschwind, on the topic of 'speed versus cost', claims that assuming 'identical organizations and the state of the art at a given time', there is a dependency of cost and speed. This might be represented qualitatively by Fig. 3-17a. This graph is often considered indicative of a situation where one tries to increase the speed of a subsystem by using faster and faster components. The limiting value is the state of the art (technically). Usually, when a configuration of equipment approaches this limit, the cost to the user becomes unreasonably high.

However, this latter situation should be further discussed to prevent a confused interpretation. If the current trend in development of equipments in the Computer-environment is examined, one realizes that often improved components are introduced to the market at lower prices to

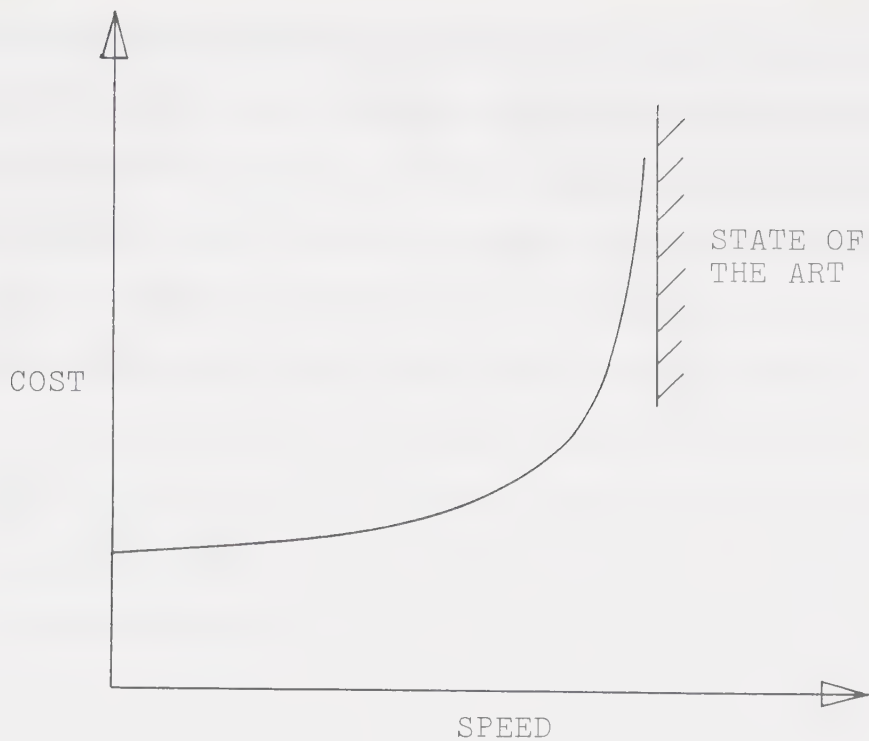


Fig. 3-17a. Design parameters speed vs cost showing general limitations imposed by state of the art.

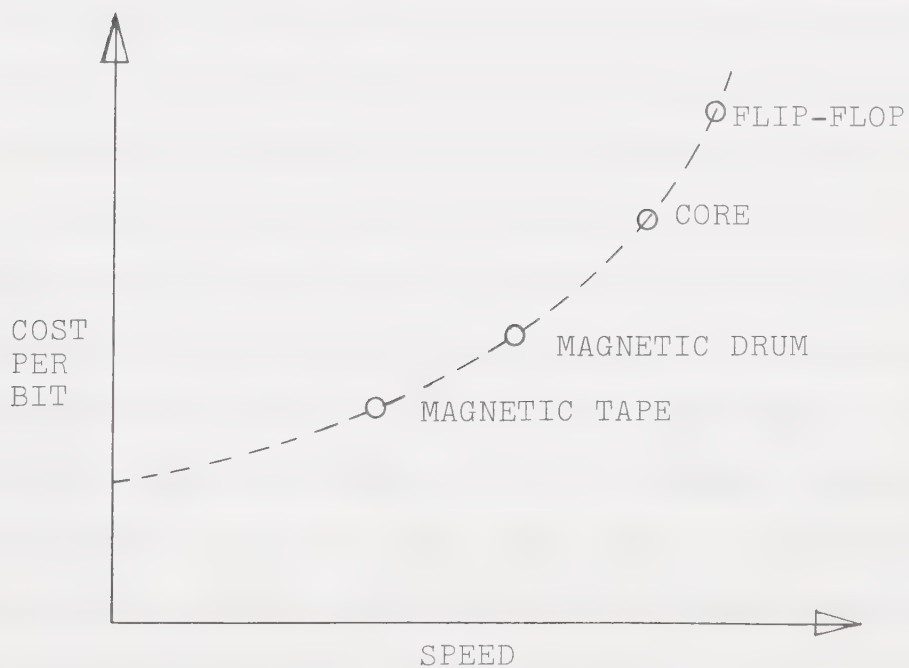


Fig. 3-17b. Speed vs equipment costs for various storage media

a user/designer than older equipments intended to perform similar functions. For example, due to advanced technology in electronic design, the costs of producing complex circuitry has progressively dropped with a corresponding increase in such parameters as speed and reliability. Table 3-1, for example, indicates how cost of producing a flip-flop (F-F) has dropped progressively with later generations of computers.

Table 3-1. Cost of F-F decreases with technical advancement

Computer Generation	I	II	III	IV
Characteristics	tubes	trans-istors	IC's	LSI's
Cost of F-F	\$100	\$10	\$5	2¢

Generally, and for purposes of developing this methodology, it is assumed that a designer selects subsystem configurations from those currently marketed. Under this assumption, then, the general shape of the curve in Fig. 3-17a is valid. For example, by using LSI components, there is generally an increase in cost for an increase in speed depending on the quality of the LSI used in the F-F.

The Design Parameters, except for Cost, are described as being positive parameters. This implies that a configuration evaluated in terms of these parameters is usually judged as 'more satisfactory' to a user as the resulting DP magnitudes increase. The DP's generally vary with cost in a manner similar to that for speed in Fig. 3-17a. It should be emphasized that Cost was used as a comparison variable throughout this study in order to orient the methodology towards providing a practical solution to the evaluation problem.

Fig. 3-17b illustrates how the cost per bit of equipment necessary

to process some storage medias varies with their attainable I/O speeds.

The Price function concept arose because of two points of view regarding the value of a subsystem's configuration. One value was that seen by a user in terms of his willingness to pay for an increase or decrease of magnitude of a design parameters. The other was that seen by a manufacturer in his costs to provide a subsystem exhibiting certain DP magnitudes. If a user/designer's willingness to pay for some magnitude of speed in a system is a linear function, then Fig. 3-18 might represent the two values discussed.

There is a certain area (shown in Fig. 3-18 as cross hatched) in which the user is willing to pay the cost of, or more than the cost of, a particular configuration. A manufacturer must be conscious that his equipment falls within this range.

Price functions analytically describe a user/designer's willingness to pay for measurable quantities of a design parameter. Converse to the definition given to Weights, these functions are intended to be non-relative measures associated with the DP's. There are only five Price functions conceivable for a subsystem, that for DP Cost is meaningless.

An example to demonstrate the practicality of the Price functions is given. Fig. 3-19 could represent a situation where a user/designer is willing to pay for an increase in speed of an arithmetic unit in a linear fashion up to a point (threshold). At this magnitude of speed, he then feels that further increase in speed is unwarranted compared to spending additional money on some other parameter, such as flexibility. This might be done by adding more storage devices. The flattening of the Price function infers this.

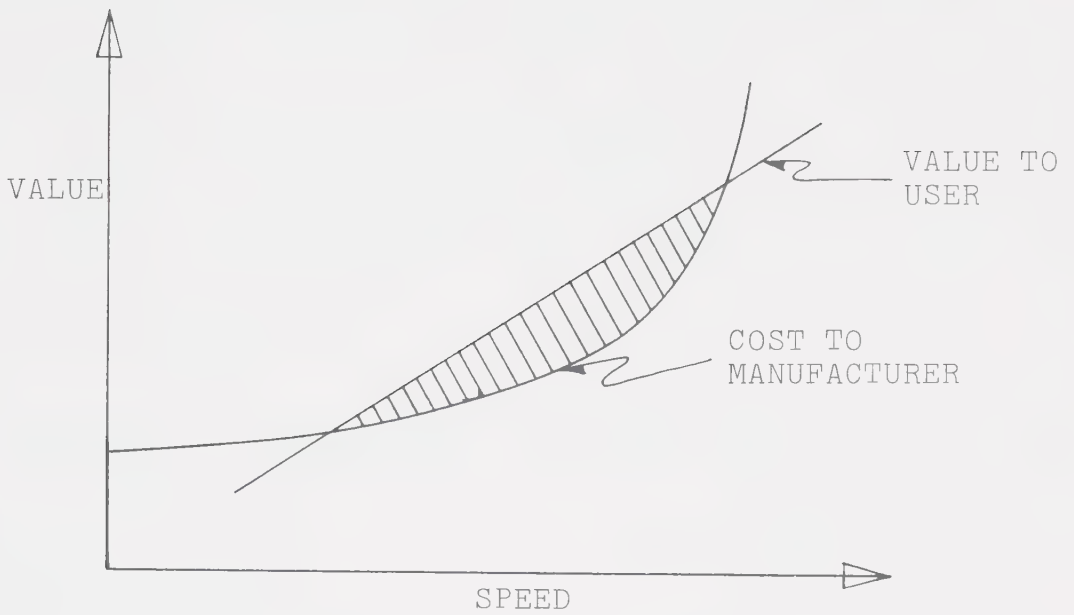


Fig. 3-18. Value of subsystem's configuration exhibiting certain speeds as seen by user and manufacturer.

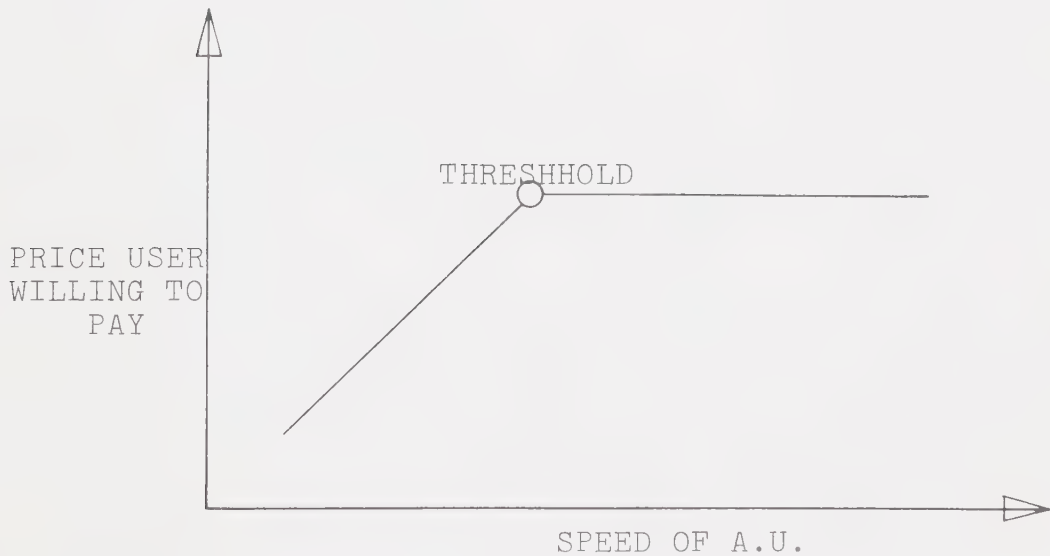


Fig. 3-19. Hypothetical price function showing user's willingness to pay for speed in arithmetic unit.

It is shown in Section 3.6 that the Worthiness index may be calculated from three user/designer described relationships, namely:

- (a) Characteristic Equations describing the DP's
- (b) Weights, providing relative importances between DP's and
- (c) Price functions, providing non-relative quantitative values of the DP's.

That is:

$$\text{Worthiness} = w(\text{Characteristic Equations, Weights, Prices}) \quad (3.4)$$

3.6 Evaluating a Subsystem

This section brings together the various concepts and definitions discussed throughout this chapter. In this way, the overall methodology for evaluating a subsystem unfolds.

It should be reemphasized that the definition of the Worthiness index (according to Equation 3.4) is wholly dependent on the characteristics of the user in the Man-environment, as well as the application undertaken. Furthermore, the preciseness of the definition is dependent on a collaboration of research findings in the many disciplines involved with study of the Man-environment. Paramount among these are the Human Factor researchers in ergonomics.

Subsequent to the difficult task of defining the elements of Eqn. 3.4, the evaluation technique is applicable in a straight forward manner. The overall subsystem evaluation keys around two main operations:

- (a) the determination of the Worthiness index and
- (b) the systematic adjustment of the subsystem configuration.

The operation described by (a) involves calculating the index as described

in Section 3.3. In particular, Worthiness was algebraically calculated according to Eqn. 3.5:

$$\text{Worthiness} = \sum_{i=1}^5 \frac{W_i P_i DP_i}{W_6 DP_6} \quad (3.5)$$

where the symbols W, P and DP correspond to the absolute magnitudes of Weights, Prices and Design Parameters resulting from analysing a particular subsystem. The subscripts represent those variables corresponding to the six parameters defined in Section 3.5 with Cost being subscript 6.

The operation described by (b) requires the adjustment of a subsystem's configuration in such a way that a better Worthiness index may result from a subsequent analysis of the new configuration. The necessary rules by which this adjustment is accomplished must be described by the user/designer in much the same way as the terms defining the Characteristic Equations of the DP's are defined. The logic of a reconfiguration algorithm must work from a data base defined from two sources:

- (a) performance variables concurrently resulting from analysing a subsystem to determine its Worthiness (for example, queue information in the Computer subsystem)
- (b) elements of the R - Space established as potential devices, and their characteristics, from which configurations may form.

In Fig. 3-20, illustrating the complete evaluation process, segmented lines depict the two information sources described above.

In a real evaluation, as is considered in Chapter IV, only a portion of the subspace of the R - Space corresponding to a specific subsystem is considered during the iterative process. In order to make the technique a practical one, only those elements of the R - Space deemed of interest

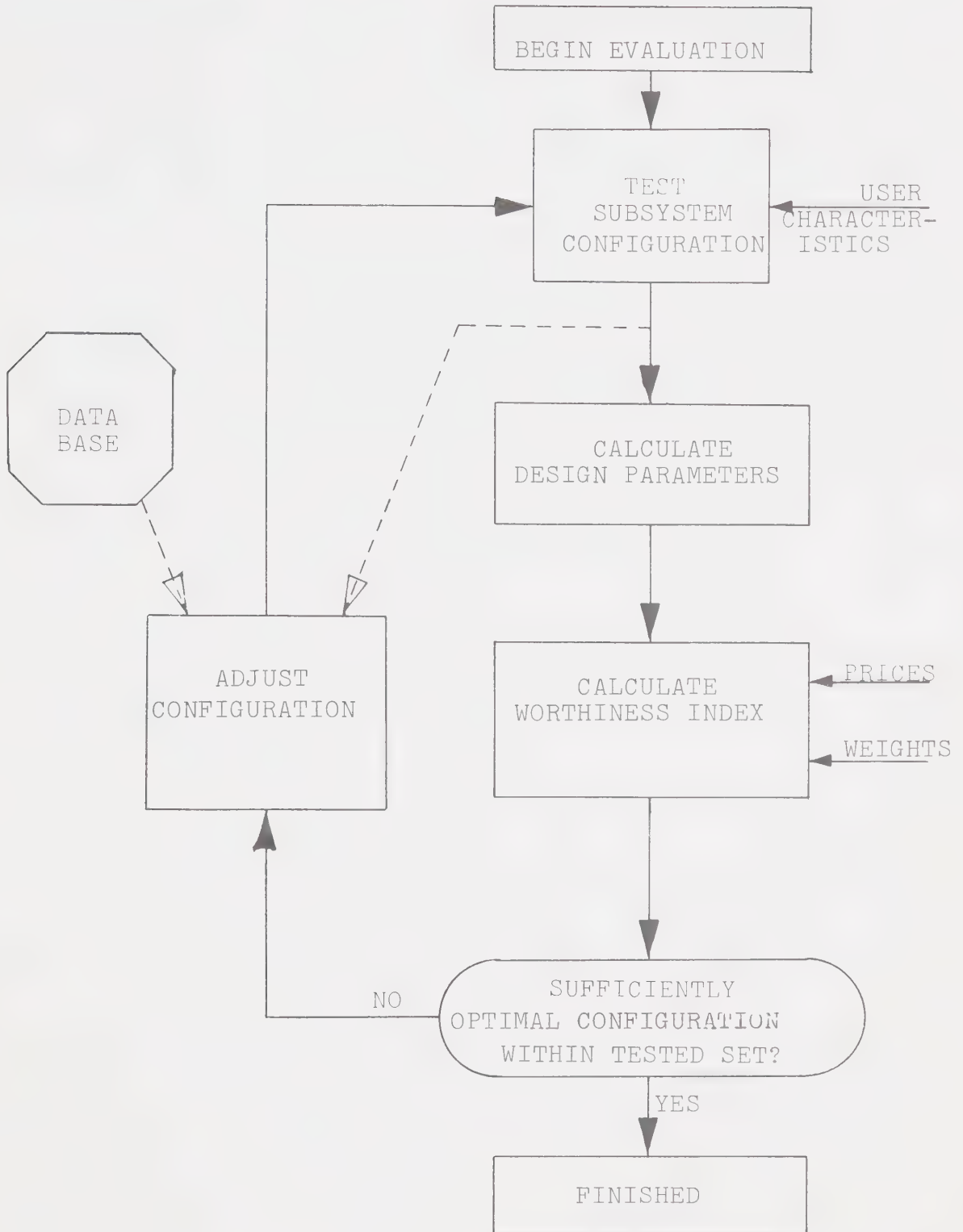


Fig. 3-20. Flow chart illustrating iterative process of evaluating a subsystem.

and/or importance by a user/designer are considered. In this respect, the complexity, or difficulty, in defining Characteristic Equations and rules for configuration adjustments of a subsystem is probably of the same order of magnitude as the size of the data base describing the potential configurations.

Then by applying reconfiguration rules upon an appropriate data base and successively testing the configurations (ideally in a real application oriented environment with real equipment), optimal solutions can conceivably be found. By nature, the speed of convergence (if a solution exists) could be dependent upon the rules applied for reconfiguration and the equations used to calculate the Worthiness index. It may be that the data base is so complicated that a mixture of linear and nonlinear relationships exist, restricting the type of process to one such as Monte Carlo.

An example illustrating the calculations necessary to find Worthiness indices for hypothetical configurations, user characteristics and applications is given. In this example, only three design parameters are deemed meaningful, these identified as accuracy (A), response (S) and cost (C). That is, only the weights, W_a , W_s and W_c are none zero according to the definition given in Equation 3.3. The prices corresponding to A and S are labelled P_a and P_s respectively.

For this hypothetical problem, the following information is necessarily defined by a user/designer:

- (a) Subsystem: Computer
- (b) Method of obtaining performance variables: simulation
- (c) Design Parameters used: A - accuracy
S - response
C - cost

(d) Weights: $W_a = 0.3$

$$W_s = 0.7$$

$$W_c = 1.0$$

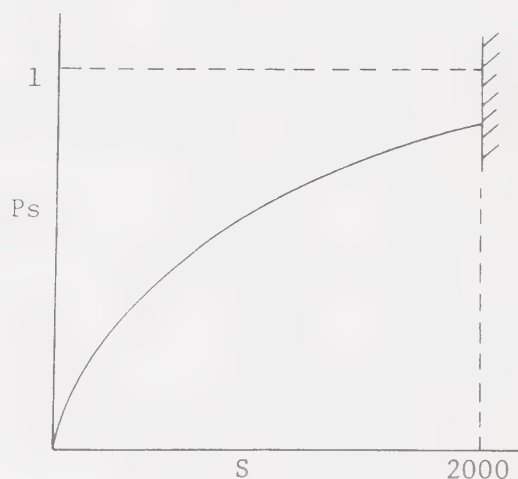
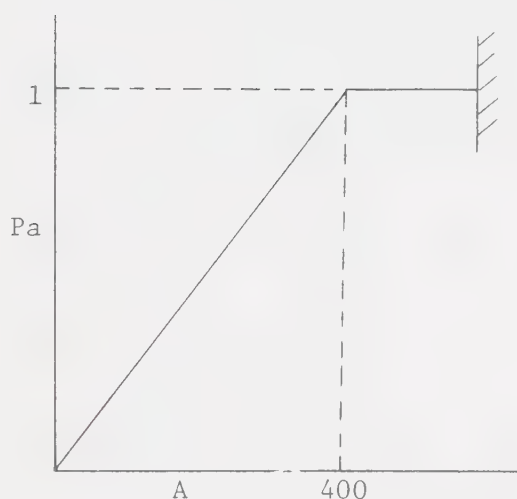
(e) Prices: $P_a = A / 400$

when $A \leq 400$

$$P_a = 1.0$$

when $400 < A \leq \text{state of the art}$

$$P_s = (1 - e^{-S/1000})$$



To calculate the Worthiness index, the following equation was used:

$$\text{Worthiness} = \frac{A \cdot W_a \cdot P_a + S \cdot W_s \cdot P_s}{W_c \cdot C} \quad (3.6)$$

Corresponding to three configurations of the subsystem, Worthiness indices for hypothetical values of accuracy, response and cost are shown in Table 3.2. These results were also plotted and illustrated in Fig. 3-21. In this simple example, configuration Y would represent the best configuration compared to X and Z.

It is imperative to point out that the absence of unit definitions

CONFIG- URATIONS	A	S	C	WORTH- INESS
X	100	300	1	66
Y	100	400	1.5	70
Z	200	300	2	39

Table 3.2. Worthiness indices resulting from three Computer subsystem configurations.

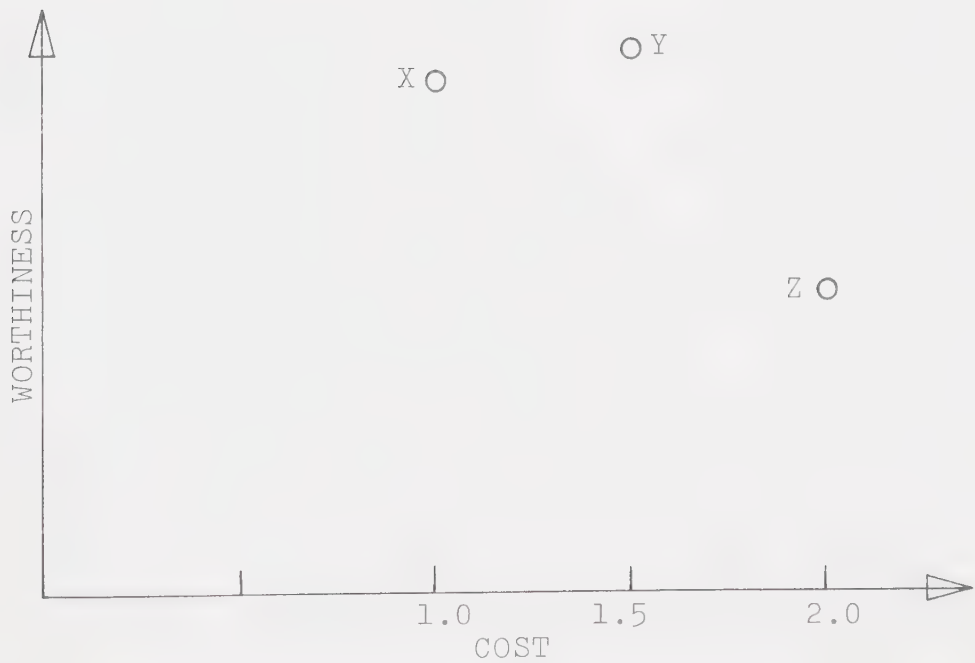


Fig. 3-21. Worthiness vs cost graph showing hypothetical configuration evaluation results.

for the design parameters used in the above example did not restrict the evaluation technique from finding an optimum configuration. Indeed, such an absence leads to a unitless definition for the Worthiness index. However, since the methodology suggests a search for relatively better configurations, absolute magnitudes for the index are unnecessary.

3.7 Limitations of the Methodology

The intent of the methodology presented in this thesis is to introduce an evaluation technique that can be applied equally well to the three subsystems defined as the Computer-environment. It is based on the assumption that by finding optimal, or close to optimal, configurations for the three subsystems, the total Computer-environment may be considered to be composed of optimal equipments.

The methodology is limited in usefulness at this time by the adequacy of user/designer's to measure the DP's by way of their Characteristic Equations. A problem arises from the very large number of interrelated performance variables of subsystems and a general lack of complete understanding among the many researchers of M-C systems regarding how to measure the effects of performance variables on human users.

The DP's used in this study were selected after researching various M-C applications currently planned or in operation. Similarly, the definitions of basic functional requirements at the Interface (Section 3.4.2) were arrived at from an analysis of a number of applications currently used. The accuracy and completeness of these ideas then must be limited by the amount of background research permissible in this study.

In the following chapter, an experiment intended to validate this methodology is described. In it, a Computer subsystem is evaluated.

CHAPTER IV

THE VALIDATION

4.1 Validation and Application of the Methodology

Chapter III described an evaluation methodology applicable to Man-computer (M-C) systems. In particular, the methodology proposed a way of evaluating each of three subsystems considered as prime components of the Computer-environment. This environment was described as including all hardware and software found on the computer side of the Interface.

It now remained to test the proposed methodology to establish some degree of confidence that it is in fact a solid proposal and one which is credible. Such a test was referred to as a validation. In this chapter, the design of an experiment for testing is described as well as details given on the particular experimental setting selected.

Section 4.2 overviews the general prerequisites needed to evaluate a subsystem according to the proposed methodology. Following this, the Nielsen model is described as a justifiable model with which to represent the Computer subsystem in the experiment. The validation was advanced as a test using this model and details are given on selecting Weights, Price functions and Characteristic Equations corresponding to certain design parameters. The experiment involved the iterative process described in Fig. 3-20 whereby the simulator was configured with various hardware device arrangements and run. Subsequent performance variables were monitored and Worthiness indices calculated. The final section of this chapter presents relevant details about the results of the experiment.

The test was designed and run in an on-line mode using the GRID [20]

system at the University of Alberta. In this way, not only was the validation completed but a real M-C system application was demonstrated.

4.2 Requirements of an Experimental Setting

In order that a particular design may be decided upon from which to test the methodology, it is important that the requirements of an overall subsystem's evaluation be classified. With reference to Fig. 3-20, and the description of its elements contained in Chapter III, the following are considered as these requirements:

- (a) An assemblage, or test system, must be available and capable of representing adjustable configurations of equipments to represent the subsystem. Such an assemblage may be in the form of real equipments or else in modelled form to represent real equipments. The former of these, although most desirable from an accuracy point of view, is in most cases considered impractical. Modelled subsystems, on the other hand, are susceptible to producing biased results due to inherent limitations characteristic to analytical and simulated models.
- (b) Characteristic Equations must be defined which relate various performance variables, resulting from analysing a subsystem's configuration, to design parameters. These equations would normally be described by a user/designer to reflect his ideas of sensitivity of, and interrelations between, various performance variables deemed important.
- (c) A set of rules must be established in order to describe the iterative process whereby a subsystem's equipments can be re-

configured to produce a more optimal configuration. In Fig.3-20, this operation is illustrated by the 'Adjust Configuration' box.

- (d) A subset of the R - Space must be described for the purpose of being used as a data base by the reconfiguration algorithm. This base must represent those devices, and their characteristics, available to form possible configurations.
- (e) An optimality criteria must be decided upon. This is essential in order to 'decide' when the iterative process has advanced far enough to enable a particular configuration (not necessarily the last tested) to be considered optimal.

In the experiment used for this validation, requirements (c) and (e) above were performed through human intervention. That is, the evaluation made the decisions as to what new configurations should be tested and when the iterative process had advanced far enough. These decision responsibilities could, of course, be handled by computer programmed logic.

4.3 The Experimental Setting

The application of the evaluation technique to a model representing the Computer subsystem is discussed in this section. The reasons for choosing this subsystem over the Input or Output ones was threefold:

- (a) More information regarding performance variables and their expected relationships was accessible for verification purposes on the Computer subsystem.
- (b) Characteristic Equations describing the DP's could be defined for purposes of this experiment with higher confidence by the author.

- (c) A computer program to represent the Computer subsystem exhibiting a least the minimum requirements for validation was available.

4.3.1 Description of the Test Computer Subsystem

The model used in the experiment was a computer program capable of simulating a variety of time-sharing computer systems. This simulator, written by Nielsen [28] [29] at Stanford University (1967), was designed for purposes of investigating certain responses in a time-sharing system caused by various hardware and software changes. The model has a general purpose design and was used to 'assist in the design and development of new time-sharing algorithms or techniques'.

In order to serve as a tool for investigating the behavior of complex time-sharing systems under a variety of conditions, Nielsen designed the model to meet three basic requirements:

- (a) The model is responsive to all changes in configurations, not only in terms of the number of devices, but also in terms of the capability of these devices. Further, the configurations are adjustable, by appropriate modifications to parameter values.
- (b) The model serves as a test vehicle for new or modified algorithms, such as for memory allocation or task scheduling. Thus, it is responsive to changes in these routines. Further, key algorithms are relatively isolated from the rest of the model to enable changes to be readily programmed and incorporated.
- (c) The model serves as a means of determining the effects of various job mixes and loads upon system performance as well as the effects of the system upon the throughput of particular job types. Thus,

it is responsive to adjustments in the requirements of particular jobs. Further, the job stream is adjustable by the change of a few parameters.

The level of detail exploited in this model was generally arrived at from the following conditional requirements:

- (a) use of FORTRAN II source language
- (b) need to model a paging environment
- (c) desire to minimize both core storage and execution time required in a host machine while still producing relevant information about the simulated system.

4.3.2 The Data Base Used for Evaluation

The concept of the R - Space was introduced in Section 3.4 and described as being composed of three partially intersecting subspaces, each corresponding to a subsystem. The validation experiment was concerned with only that subspace associated with the Computer subsystem and illustrated by the single cross hatched area in Fig. 4-1.

As was discussed in Section 3.6, the actual size of the 'C' space used as a data base by the evaluation technique is related to the underlying complexity of the subsystem model being evaluated. In the case of Nielsen's model, the double cross hatched area of Fig. 4-1 might represent the effective elements of the R - Space from which the evaluation process may select or adjust configuration. This size is related to the level of detail modelled.

For example, Nielsen's model does not permit adjustments of a data channel-CPU memory reference interference factor, or the adjustment of

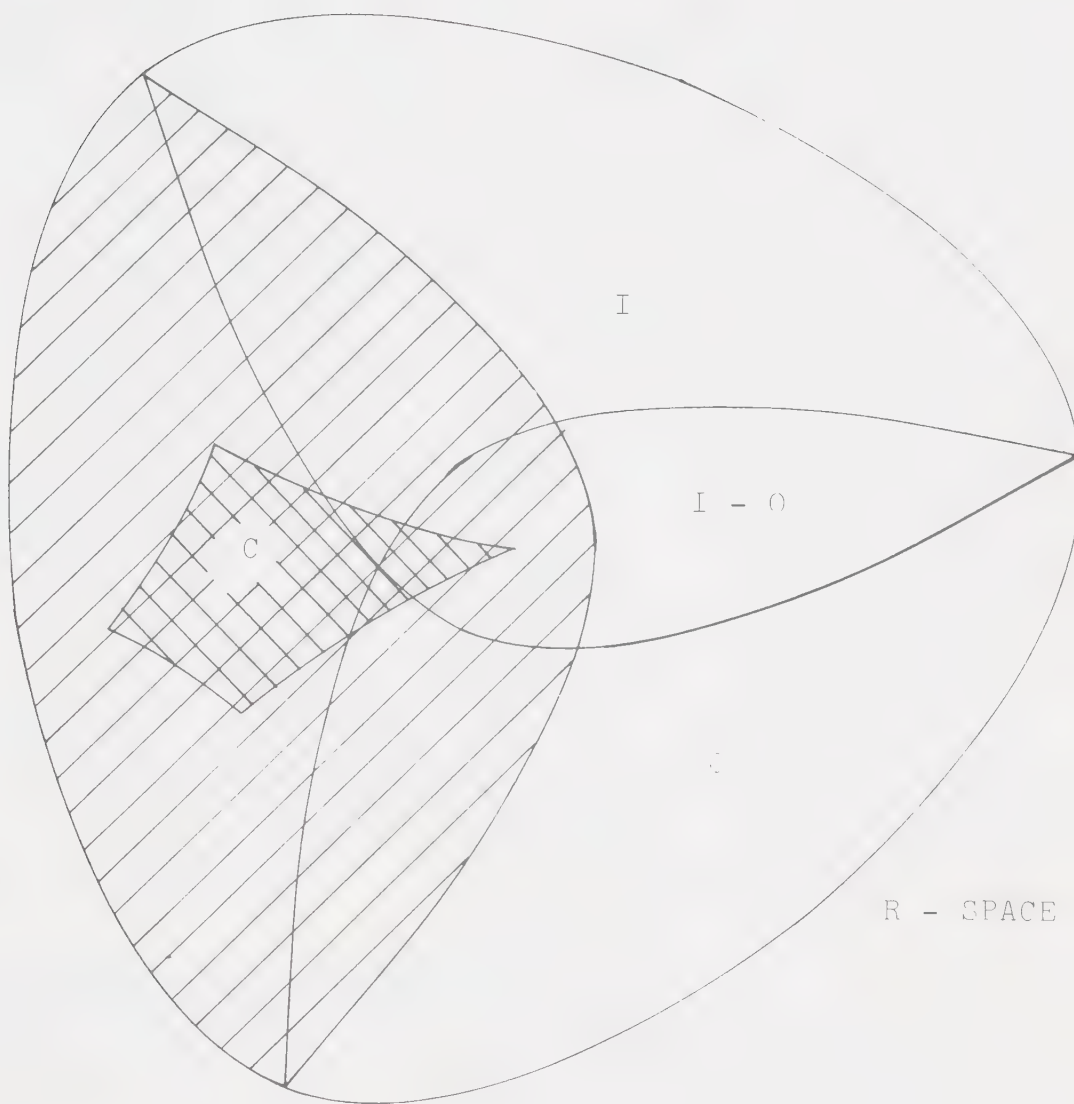


Fig. 4-1. Nielsen's model concerns only equipments found in double cross hatched area of R - Space.

word sizes within the modelled IBM 360/67. However, more gross factors such as the ability to adjust core sizes or the number and types of I/O devices are available. 'Time factors', such as interactive rates and response times seen at on-line terminals are of concern in the model, but those such as basic machine cycle times or memory reference times are not modelled. For complete details on the capabilities of this model, beyond those mentioned here as examples in support of the validation, refer to [28].

In the experiment, concern was given to only those elements of the R - Space considered as hardware devices, and their characteristics, associated with Nielsen's model. Table 4-1 illustrates the primary devices, characteristics of these devices, types originally modelled in Nielsen's simulator and upper limits on the number of device types permitted. Of course, since a device is completely defined by its characteristics in this model, the reference to 'types modelled' has no functional bearing beyond a familiarity flavour, i.e., IBM type equipment. For this same reason, these characteristics were not changed in the experiment with the intent of testing the behavior of real equipments as much as possible.

4.4 Choosing Characteristic Equations

Having satisfied two of the requirements described in Section 4.2, it remained to define a set of Characteristic Equations precluding the actual experimentation. Such a set was necessary to resolve associations between various performance variables obtainable by monitoring Nielsen's model as it operated.

The simulator itself set the bounds within which performance variables could be selected for monitoring and eventual calculation of certain

Table 4-1 Devices and their characteristics considered as data base for evaluation using Nielsen's model.

DEVICE	TYPE MODELLED	NUMBER POSSIBLE	CHARACTERISTICS
CPU	IBM 2067	4	None
CORE	IBM 2365	ANY	Number non-system activity pages
PAGING DRUM	IBM 2301	4	Capacity reserved for priority pages Total Capacity in pages Revolution time Beginning slot number Total number slots Cutoff time
PAGING DISK	IBM 2314	4	Total Capacity in pages Minimum seek time Average seek time Average rotational delay Time to transmit single page Number independent access arms
TERMINAL	IBM 2741	234	First job type First job start time Inter job wait distribution Probability of selecting next job type
DISK	IBM 2311	TOTAL 50	Channel attached to Transmit rate Minimum seek time* Average rotational delay* Time to transmit single page Average seek time*
TAPE	IBM 240x		
LINE PRINTER	IBM 1403		
CARD READER	IBM 2540		

* Not all non-paging devices have same meaningful characteristics

design parameters. The constraints imposed on this selection were much the same as those determining the size of data base used (refer Section 4.3.2). For the experiment, a rational choice of performance variables was made and Characteristic Equations derived to allow analytical measurement of three design parameters. The variables chosen were those:

- (a) considered of interest in time-sharing systems from a user's point of view.
- (b) available from Nielsen's simulator.
- (c) considered by other researchers as being relevant in system performance measurement.

The rationale used and the selection of performance variables are given:

Accuracy: Characteristic Equations for this DP were not defined for the experiment. Nielsen's model was not designed to analyse various accuracies resulting from different configurations.

Reliability: According to Nielsen, the primary aim of the model was 'to serve as a tool for advancing the general knowledge of time-sharing and for assisting the analysis ... for decision purposes'. Accordingly, the model was focused on the 'normal' state of a system and such things as hardware malfunctions were ignored. Consequently, this DP was not considered definable for the experiment.

Response: This DP was considered by monitoring the ratio of response time per raw execution time required for each interaction. Nielsen's model was capable of collecting such distributions for any application load on a configuration.

The Characteristic Equation describing Response was defined in terms of the mean and standard deviation of this ratio. It was felt that these

two statistics alone were most apparent to a user at a terminal with respect to the speed of a reply from the computer. A small mean ratio (as close to unity as possible) is usually desirable, as well as is a small standard deviation (close to zero). The tolerable limits of these two are considered user dependent in this study.

Facility: This DP was not considered relevant to the Computer subsystem in this experiment. Normally, it is describable only for the Interface equipments: the Input and Output subsystems (refer Section 3.5).

Flexibility: In this validation, Flexibility was considered to be a measure of both the number of and the types of hardware devices available to a user. Its Characteristic Equation is defined in Section 4.6.

Cost: Many of the variables available from Nielsen's model permit a variety of information to be included within this DP. The following describes three factors considered in defining a rationale Characteristic Equation.

First, basic hardware costs of equipment in the Computer subsystem were considered. The situation arising from various cross switching arrangements in a system, enabling various I/O devices to be attached to core memories via different channel and control units, needed be considered. In Nielsen's model, all paging devices were considered to be attached through dedicated controllers and channels. Consequently, their combined cost could be represented by a single figure. However, user dedicated I/O devices could be specified as being attached to any number (up to 32) channels. For purposes of illustrating the experiment, it was decided to consider all direct access (non-paging) devices as being attached to a single selector channel. Up to a point, this constraint was not considered as being unrealistic compared to real operational systems. Then,

since the number of channel and control units was not considered to vary between configurations, their costs could be split between their attached devices.

Second, the operating and maintenance (O&M) costs were considered. These costs could be interpreted in a number of ways, one being simply as a function of the number and types of devices in a configuration. If the O&M costs are considered as a linear function of the cost of a device, then these costs can be considered as absorbed within the basic hardware costs. This linearity assumption was made in the experiment.

Third, since most real-time installations charge their users according to utilization attributed by them on resources, this factor was considered. In the model, statistical collecting routines provide utilization figures for all devices of a configuration. A breakdown of those figures used in the validation experiment is given in Table 4-2.

Table 4-2 Utilizations available from Nielsen's model

DEVICE	UTILIZATIONS COLLECTED
CPU	% EXECUTION % OVERHEAD % IDLE
PAGING DRUM	% READ % WRITE
PAGING DISK	% READ % WRITE
NON- PAGING DEVICE	% READ % WRITE

In essence then, Cost was considered to be defined by two elements:

- (a) cost representing all hardware and their O&M charges.
- (b) costs to user representing his load utilization on resources.

The resulting Characteristic Equation is given in Section 4.6.

4.5 The Experimental Procedure

Nielsen's simulator, as initially designed, consisted of thirty-one subroutines, about 7000 FORTRAN source images and was intended to be run in a batch environment. When run in this mode as a set of four overlays, it was found to require about 150K of /360 core if single I/O buffers for the statistics collected were assigned. Running the model as intended by its designer, the experiment would have involved one batch run for each iteration of the evaluation technique. Following each simulator run, an off-line analysis of the results from a line printer would have been required to calculate Worthiness indices.

For purposes of the experiment, certain of the operations illustrated in Fig. 3-20 were carried out by computer programmed logic. The remaining ones were handled through human intervention. In Fig. 4-2, those blocks labelled 'C' depict those being handled by the former type; those labelled 'H' represent the latter.

Because of the distinctly separable decision tasks involved with the iterative evaluation process intended for the experiment, the practicality of using the simulator on-line was most apparent. In addition, by using this mode for experimentation, not only was a validation carried out but the application of a real M-C system was demonstrated. That is, the experiment involved an evaluation of a simulated M-C system (Nielsen's) on a real M-C system.

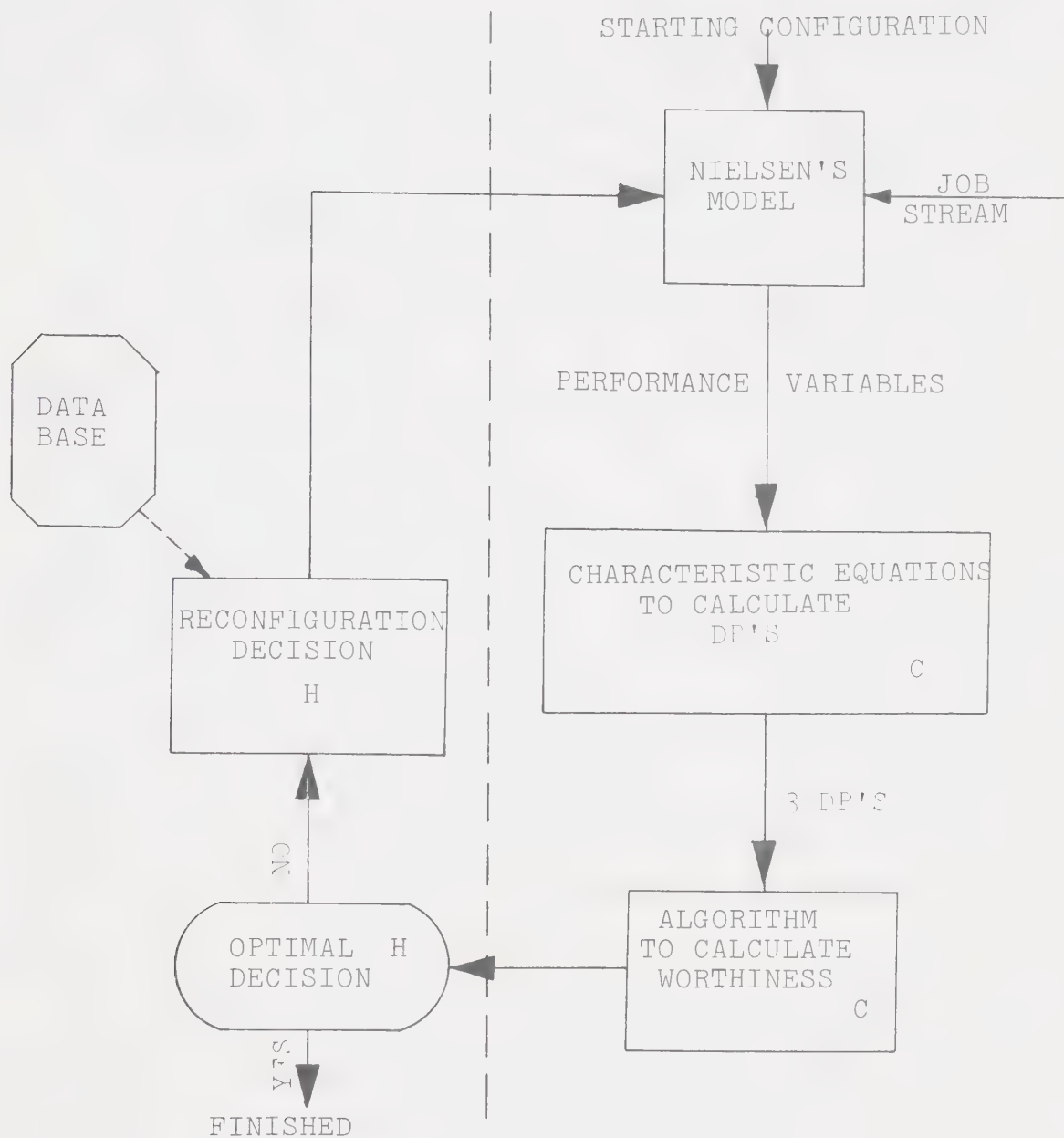


Fig. 4-2. Operations required to validate methodology by evaluating Nielsen's model.

Consequently, it was decided to modify Nielsen's model and write appropriate interfacing code (called the Simulator Monitor) to permit its use on-line. The main operational advantages apparent with this mode of operation, compared to batch, were:

- (a) Only one linkage editor step required for all iterations.
- (b) Close control of simulator possible by evaluator.
- (c) Convenient adjustment of simulated job stream characteristics and configurations.

Items (b) and (c) above are described in Appendix B along with details of operating the simulator on-line.

The experiment was designed to incorporate the human evaluator at the GRID terminal (refer to Section 3.2). This device was connected to an IBM 360/67 via an IBM 2703 transmission control unit. The host computer was operated under the MTS [26] time-sharing system. The relationship between the MTS, Simulator Monitor, Nielsen's model and the GRID program blocks, as seen by an evaluator, are illustrated in Fig. 4-3. Relevant details, as well as a source listing, of the monitor are given in Appendix A.

In the experiment, the simulated system was configured to represent the TSS Version I supervisory software (refer Nielsen [28]). The workload on the simulated system was adjusted to represent the almost trivial case of a single terminal user running a sequence of interactive jobs. In the original design, Nielsen choose to allow a terminal to select each successive job run at random. For the experiment, the model was modified to allow jobs to be predetermined for each iteration. The jobs chosen were restricted to a set of 34 which Nielsen designed to represent a wide variety

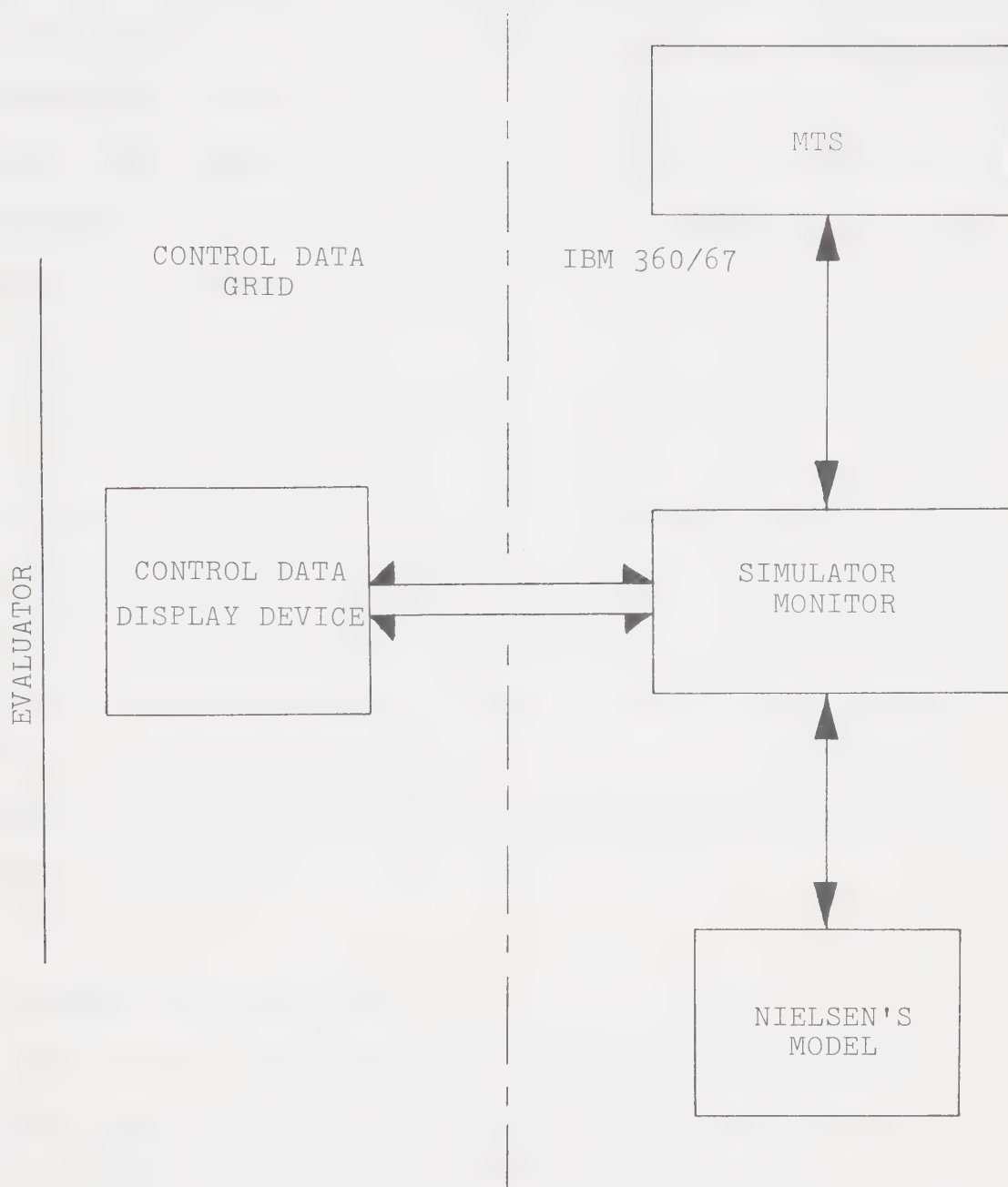


Fig. 4-3. Relationship between programs in display terminal and /360 as seen by evaluator.

of work load types which he felt described a university environment.

In order to add further realism to the experiment, the External environment was considered. A single background type job (batch) was included in each simulated run to represent a controlled and repeatable interference to the Computer subsystem. Its characteristics remained constant, allowing the relative effects of the single terminal loads to be realized.

The experiment involved evaluations on three job streams entered from the single user's terminal. Each stream represented various user characteristics, such as wait times between interactions, type of application and interjob delays. For each job stream, ten configurations were modelled and tested. In the interests of better facilitating the comparison between Cost-Worthiness plots, the same configurations were modelled for different job streams. Worthiness indices were calculated via a computer programmed algorithm reflecting the three Characteristic Equations detailed in Section 4.6.

4.6 Results of the Experiment

This section details the actual Characteristic Equations used in the experiment to define the three design parameters of interest. Also, the corresponding Weights and Price functions used are defined. The results are tabulated in this section; however, discussion of the significance of the results as well as any overall observations implied from them is reserved for Chapter V.

It was not considered relevant to include the actual performance variables extracted from every run on Nielsen's model. However, in the int-

erest of completeness, one set of variables from an actual run, plus the associated calculations required to produce a Worthiness index, are given in Appendix C.

The evaluation settings are given in much the same fashion as those given in the example in Section 3.6.

(a) Subsystem: Computer

(b) Method of obtaining performance variables: Nielsen's model

(c) Design Parameters sued: F - flexibility

R - response

C - cost

(d) Characteristic Equations:

$$F = 0.7 \text{ NDVCE} + 0.3 \text{ NTYPE} \quad (4.1)$$

where: NDVCE is number of devices (pages considered
as PAGES/30),

NTYPE is number of individual types of devices

$$R = 700 / \text{MEAN} + 300 / \text{SD} \quad (4.2)$$

where: MEAN is average ratio of response time per
execution time of all interactions,

SD is standard deviation of this ratio

$$C = \sum_{\text{all } j} \text{COSTDVCE}_j + \sum_{\text{all } j} \text{COSTDVCE}_j \cdot \text{UTILDVCE}_j \quad (4.3)$$

$$= \sum_{\text{all } j} \text{COSTDVCE} \cdot (1 + \text{UTILDVCE}_j)$$

where: COSTDVCE_j is the cost of device j ,

UTILDVCE_j is the utilization of device j

$$(e) \text{ Weights: } W_f = 0.3 \quad (4.4)$$

$$W_r = 0.7 \quad (4.5)$$

$$W_c = 1.0 \quad (4.6)$$

$$(f) \text{ Price functions: } P_f = F / 30 \quad (4.7)$$

$$P_r = R / 15 \quad (4.8)$$

(g) Worthiness function:

$$\text{WORTHINESS} = \frac{F \cdot W_f \cdot P_f + R \cdot W_r \cdot P_r}{W_c \cdot C} \quad (4.9)$$

In order to include additional realism to the experiment, the constants used to represent relative device costs (COSTDVCE's in (d) above) were chosen from actual rental charges applied to similar equipments in the Computing Center at the University of Alberta. These costs were assessed in June 1970 and consolidated in Table 4-3, along with the relative costs between devices. The figures were derived by considering the manner that configurations were assembled in Nielsen's model.

Table 4-3. Relative costs used in experiment to represent cost modifications to configurations.

DEVICE	ADDITIONAL COST TO CONFIGURE	RELATIVE COST
CPU	\$11394	1.0000
CORE PG.	\$95	.0083
PG. DRUM	\$5048	.443
PG. DISK	\$5717	.5018
TAPE	\$807	.0708
PRINTER	\$771	.0677
READ/PUNCH	\$577	.0506
DISK	\$620	.0544

The results of varying hardware configurations on different job streams are listed in Tables 4-4 through 4-6. Included in these tables are not only Worthiness figures and the corresponding costs, but also the resulting magnitudes of response (R) and flexibility (F). In this way, changes apparent in Worthiness figures may be viewed with respect to alterations in the three design parameters between iterations.

The last two columns of these results were plotted as Cost versus Worthiness. These are illustrated by Fig. 4-4 through 4-6. On both axes, a truncated scale permits clarification in viewing the trend caused by different configurations. Corresponding to each of the plotted points, a letter identifies the corresponding configuration (RUN) number seen in the supporting tables. This was also used to permit more convenient comparisons between plots.

On both the result plots and their associated tables, the job sequence numbers correspond to the jobs designed by Nielsen in his original model version. Each job type is described in [29].

RUN	HARDWARE CONFIGURATIONS								RESULTS			
	CPUS	CORE (PAGES)	PAGE DRUMS	PAGE DISKS	TAPES	PRIN- TERS	READ/ PUNCHS	DISKS	F	R	C	WORTHINESS
A	1	166	2	1	4	3	2	4	18.17	9.66	54.62	.1403
B	1	166	2	2	4	3	2	4	18.87	10.63	55.07	.1603
C	1	166	1	2	4	3	2	4	18.17	9.45	54.04	.1383
D	1	166	1	1	4	3	2	4	17.47	8.38	45.10	.1403
E	1	102	2	1	4	3	2	4	16.68	9.63	59.54	.1193
F	1	50	2	1	4	3	2	4	15.47	9.61	59.11	.1133
G	1	102	2	1	2	3	2	4	15.28	9.63	59.40	.1121
H	1	102	2	1	2	2	0	2	11.48	9.63	57.12	.0987
I	1	50	1	1	1	1	1	1	8.47	8.37	52.02	.0766
J	2	102	1	1	1	1	1	1	10.38	8.38	98.90	.0440

Table 4-4. Experimental results for application job stream (8,10,12,14)

RUN	HARDWARE CONFIGURATIONS								RESULTS			
	CPUS	CORE (PAGES)	PAGE DRUMS	PAGE DISKS	TAPES	PRIN-TERS	READ/PUNCHS	DISKS	F	R	C	WORTHINESS
A	1	166	2	1	4	3	2	4	18.17	9.77	61.05	.1270
B	1	166	2	2	4	3	2	4	18.87	10.67	55.42	.1601
C	1	166	1	2	4	3	2	4	18.17	9.45	54.04	.1383
D	1	166	1	1	4	3	2	4	17.47	8.38	53.54	.1182
E	1	102	2	1	4	3	2	4	16.68	9.76	60.52	.1195
F	1	50	2	1	4	3	2	4	15.47	9.74	58.08	.1174
G	1	102	2	1	2	3	2	4	15.28	9.70	54.34	.1238
H	1	102	2	1	2	2	0	2	11.48	9.70	52.56	.1081
I	1	50	1	1	1	1	1	1	8.47	8.37	37.96	.1050
J	2	102	1	1	1	1	1	1	10.38	8.38	98.90	.0440

Table 4-5. Experimental results for application job stream (30,13,2,2,10)

RUN	HARDWARE CONFIGURATIONS								RESULTS			
	CPUS	CORE (PAGES)	PAGE DRUMS	PAGE DISKS	TAPES	PRIN- TERS	READ/ PUNCHS	DISKS	F	R	C	WORTHINESS
A	1	166	2	1	4	3	2	4	18.17	9.59	59.88	.1268
B	1	166	2	2	4	3	2	4	18.87	10.62	54.61	.1616
C	1	166	1	2	4	3	2	4	18.17	9.46	54.04	.1383
D	1	166	1	1	4	3	2	4	17.47	8.38	45.10	.1403
E	1	102	2	1	4	3	2	4	16.68	9.59	59.34	.1192
F	1	50	2	1	4	3	2	4	15.47	9.57	57.91	.1151
G	1	102	2	1	2	3	2	4	15.28	9.61	53.49	.1242
H	1	102	2	1	2	2	0	2	11.48	9.58	48.11	.1165
I	1	50	1	1	1	1	1	1	8.47	8.37	37.96	.1050
J	2	102	1	1	1	1	1	1	10.38	8.38	98.90	.0440

Table 4-6. Experimental results for application job stream (14,15,16,17)

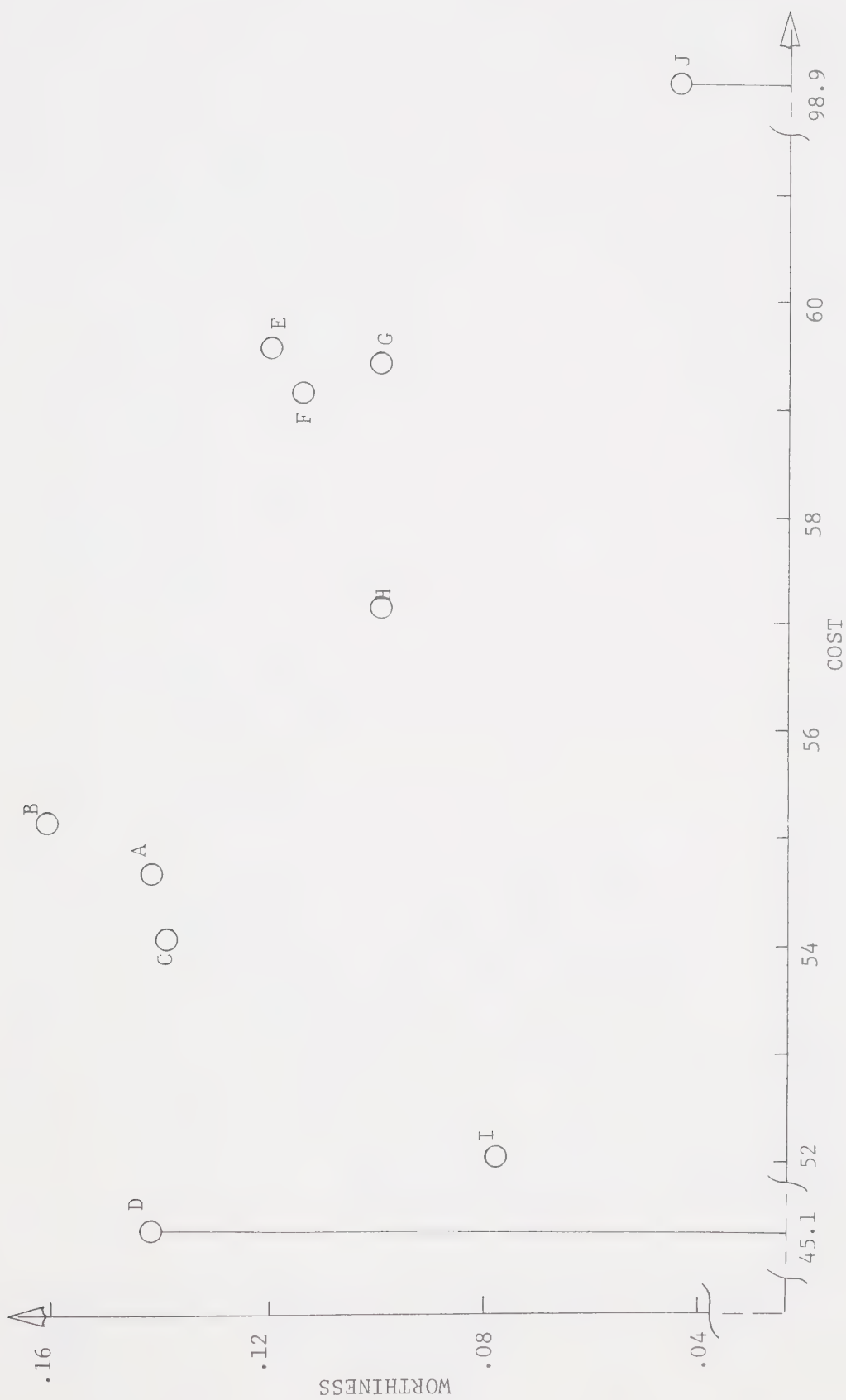


Fig. 4-4. Plot of Worthiness versus Cost for job stream (8,10,12,14)

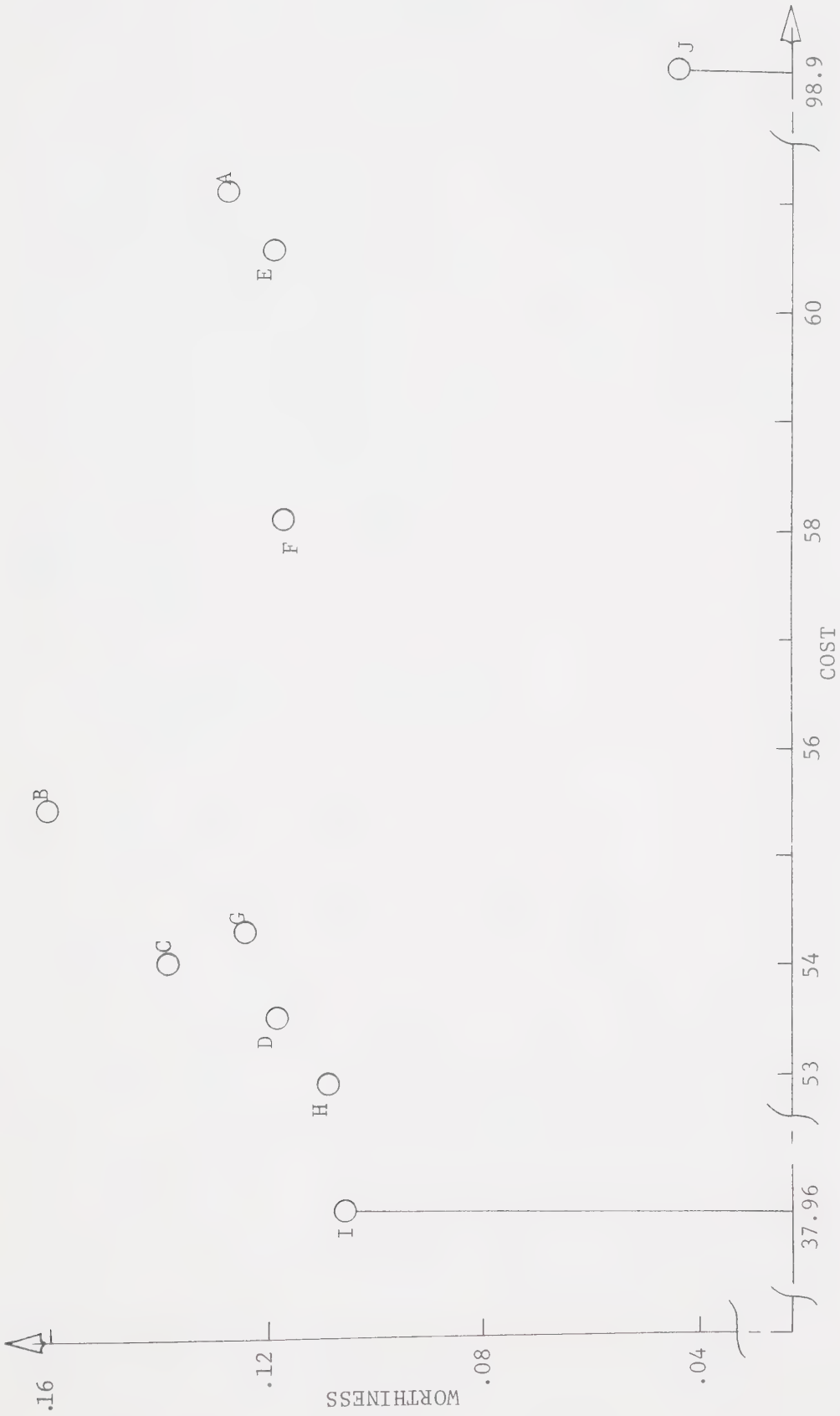


Fig. 4-5. Plot of Worthiness versus Cost for job stream (30,13,2,10)

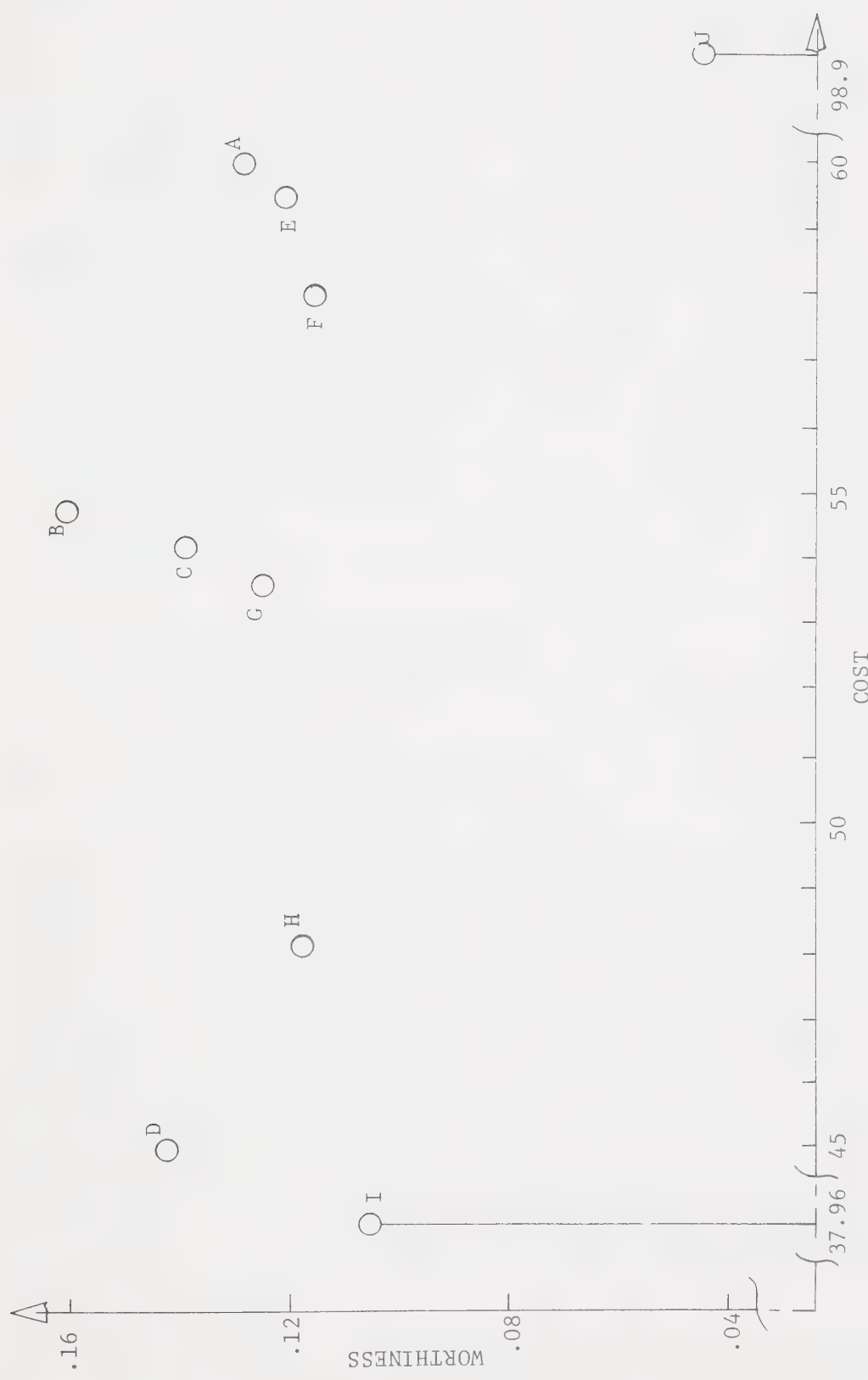


Fig. 4-6. Plot of Worthiness versus Cost for job stream (14,15,16,17)

CHAPTER V

CONCLUSIONS

5.1 Summary and Conclusions

The primary object of this research was the development of a methodology with which to evaluate M-C systems. Research was concentrated on the computer side of the Interface. As part of the validation process, the methodology was used to evaluate an experimental application. Two sets of conclusions are presented in this section, those relating to: the development of the methodology, and the evaluation of a subsystem using the methodology.

5.1.1 Development of the Methodology

The methodology was conceived to be independent of the application and the subsystem analyzed in the Computer-environment. Its implementation was concentrated on first defining a relatively unique and independent set of terminologies called Design Parameters. The many qualities commonly attributed to M-C system performance conceivably fall within this set. The technique hinges on the operational definitions of these parameters by way of Characteristic Equations which relate performance measures of subsystems to Design Parameters.

The methodology involves evaluating a subsystem by systematically adjusting its configuration and measuring Worthiness indices for each configuration. By relative comparison between these indices, an optimal point is identifiable.

Whether an acceptable optimal point results largely depends on the

size of the data base defined. This is dependent on the grossness or fineness of detail exhibited by the tested subsystem whose elements are contained in the R - Space. Also, user/designer selected Weights and Price functions influence where and if an acceptable optimal occurs.

The operations necessary for an evaluation of a subsystem could be automated by computer programmed logic. Current M-C systems are built by designers using certain rules defining 'good' configurations; therefore, logic rules do exist from which to base such judgements on as adjusting configurations and deciding when the iterative process of evaluation has advanced far enough.

The seemingly broad definitions given the DP's are regarded as pointing the direction needed towards fully identifying the variables within these terminologies. This study advances the DP's as being those necessary for evaluating any subsystem from a single user/designer's point of view.

5.1.2 Validation of the Methodology

The validation was projected as a general test on the methodology by analysing the Computer subsystem. Similar models of the Input or Output subsystems could, if available, have been used for the experiment. However, a considerable amount of additional research would have been necessary to obtain adequate knowledge required to define reasonable Characteristic Equations. For example, an adequate definition of design parameter Facility would be a most difficult task.

In the experimental setting, the data base was only part of that available for adjustment in the model used. For a more complete analysis of time-sharing systems, both hardware and software need be considered.

Obviously, how devices are utilized is as much dependent on scheduling and resource allocation algorithms as on hardware devices available.

The Characteristic Equations defined for the experiment demonstrate the application of the methodology in a realistic type environment as well as the ability to obtain conceivable Cost-Worthiness relations. The definitions used for the DP's were minimal for purposes of validation. More realistic charging schemes likely include such variables as on-line times and file storage per unit time magnitudes. Also, most device/resources are not charged for as linear functions of their quantities, as was assumed in the experiment.

For the validation, successive configurations were assembled during the iterative process by a random selection of possible resources. This was sufficient to demonstrate the evaluation technique. However, with a larger data base, the reconfiguration algorithm would necessarily analyse the performance variables from the tested subsystem. In this way, the iterations required to find an optimal configuration could be minimized.

The experimental results, when plotted as Cost versus Worthiness, illustrated the effective usefulness of various configurations under different application loads. Figs. 4-4 through 4-6 show that not only do optimal Worthiness points appear in these plots, but that the positions of these points vary between job streams. Furthermore, the rate with which Worthiness figures decrease from these optimal points is varied.

Conceivably, a user/designer could view a similar plot, corresponding to his application and user characteristics. From these, he would be able to decide if:

- (a) given a particular configuration, how good is it compared to an optimal one for the same application?

- (b) given a certain amount of purchasing capital, with which to buy a configuration, how close to the optimal will a configuration be and still be within the capital constraints?

It may be that by a relatively slight increase in this capital resource, a much improved configuration is possible.

The results of the validation establish the methodology as being a practical approach to M-C system evaluation.

5.1.3 Uses of the Experimental Setting

The on-line experimental setting described in Chapter IV has application beyond that of providing an acceptable validation setting. In conjunction with this research, the simulator monitor was designed to be used as a study tool to advance the general knowledge of the use of models of M-C systems. As such, many additional options were included which were not directly used during the validation. To allow research to be extended in this direction, these options and their use are included in Appendix B.

5.2 Directions for Further Research

It is clear that this research raised many questions. While some have already been discussed with the goal of future research, this section attempts to bring together those felt relevant, specify some of the important questions raised and indicate those directions not yet specified.

In order to further test the methodology, the 'most critical' subsystems of the Computer-environment must be further analysed. These are the Input and Output subsystems at the Interface. This requires adequate

models with which to test various configurations subject to realistic applications.

Prior to this, however, research must be directed to explicitly identify those characteristics of the Man-environment which are affected by variables changing in the Computer-environment. What are these variables and to what degree are they sensitive to changing the flow of useful information in the closed M-C loop?

The experimental setting used in this study was not selected to be typical of all relevant settings. Further testing under experimental conditions which consider both hardware and software are required to further the validation process. This could be extended using Nielsen's model.

Throughout this study, it has been suggested that an optimal Computer-environment might be one consisting of optimal subsystems. Therefore, the methodology concentrated on evaluating the subsystems on a basis independent of one another. However, whether this additive property is in fact a worthy assumption needs to be examined.

To further the research undertaken in this study, a mathematical approach to evaluating M-C systems could be advanced. With such techniques, analytical models could be developed with which formal mathematical rules could be applied. For example, a step forward would be a mathematical method of estimating a maximum Worthiness index possible from a particular data base representing devices/characteristics of some subsystem.

Finally, research could be further directed towards the monitoring of real M-C systems as they operate. In this way, statistics could be collected and used as realistic input data to other evaluation models. Also, from such observation, better identification of the Interface variables and their sensitivities affecting man-machine interaction could be gained.

BIBLIOGRAPHY

1. Baker, J.D. "From the diet of worms to the bucket of worms: a protest concerning existing display dogma for information systems", Second Congress on the Information System Sciences, 1965, pp. 429-432.
2. Bohling, D.M. and L.A. O'Neill. "An interactive computer approach to tolerance analysis", IEEE Trans. Computers, 1970, vol. C-19, no. 1, pp. 10-16.
3. Carbonell, J.R. "On man-computer interaction: a model and some related factors", paper presented at IEEE systems Science and Cybernetics Conference, 1967.
4. Chapanis, A. "Man-machine engineering", Tavistock Publications, Ltd., London, 1968.
5. Davis, M.R. and T.O. Ellis. "The RAND tablet: a man machine graphical communication device", Information Display, 1967, vol. 4, no. 4, pp. 85-90.
6. Davis, R.M. "Man-machine communication", Annual Review of Information Science and Technology, 1966, vol. 1, pp. 221-254.
7. Debons, D. "Quantitative evaluation of displays", Third Congress of Information System Science and Technology, Thompson Book Company, Washington D.C., 1967, pp. 423-428.
8. Dennis, J.B. and E.L. Glaser. "The structure of on-line information processing systems", Third Congress on Information System Sciences, Thompson Book Company, Washington D.C., 1967, pp. 5-14.
9. Devere, G.J., G. Hargreaves and D.M. Walker. "The DAC-I system", Datamation, June 1966, vol 12, no. 6.
10. Digital Computer Graphics, American Data Processing, Inc. Detroit, Mich. vol. 2.
11. Dolotta, T.A. "Functional specifications for typewriter-like time-sharing terminals", Computing Surveys, 1970, vol. 2, no. 1.
12. Evans, J.A. "A methodology for command information system analysis", Third Congress on Information System Science and Technology, Thompson Book Company, Washington, D.C., 1967, p. 251.
13. Foley, J.D. "Evaluation of small computers and display controls for computer graphics", Computer Group News, Jan-Feb 1970, pp. 9-21.

14. Gold, M.M. "A methodology for evaluating time-shared computer system usage", dissertation, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1967.
15. Grimberg, J.C. "A parametric approach to the evaluation of military information systems", Technical Documentary Report, Automatic Data Field Systems Command, Bunker-Ramo Corp. McLean, Virginia, 1966.
16. Gschwind, H.W. "Design of digital computers: an introduction", Springer-Verlag, New York, 1967.
17. Hubbarth, W.F. "Shrinking the man-computer interface", Control Engineering, Aug 1965, p. 63.
18. Jacks, E.L. "The DAC-I: a computer system for the study of graphical man-machine communication", General Motors Engineering Journal, Second quarter 1965, vol. 12, no. 2, pp. 2-8.
19. Jacobs, J.H. and T.J. Dillon. "Interactive saturn flight program simulator", IBM Systems Journal, 1970, vol. 9, no. 2.
20. Jacobsen, F.B., K.F. May and J.P. Penny. "Computer graphics for the FORTRAN programmer", Reference manual for the IBM 360/GRID graphics software system, Dept. of Computing Science, University of Alberta, 1970.
21. Jones, M.M. "On-line simulation", Proceedings - ACM National Meeting, 1967, pp. 591-600.
22. Kern, J.L. "The computer-operator interface", Control Engineering Sept 1966, p. 118.
23. Martin, J. "Design of real-time computer systems", Printice-Hall, Inc. Englewood Cliffs, N.J., 1967.
24. Meeker, R.J., et al. "Updating some ground rules for man-machine simulation", System Development Corporation, Santa Monica, California, 1968.
25. Mills, R.G. "Man-machine communication and problem solving", Annual Review of Information Science and Technology, 1967, vol. 2.
26. Michigan Terminal System. University of Alberta Computing Center Publication, 1970.
27. Nickerson, R.S. "Man-computer interaction: a challenge for human factors research", IEEE Trans. Man-Machine Systems, Dec 1969, vol. MMS-10, part II, pp. 164-180.

28. Nielsen, N.R. "The analysis of general purpose computer time-sharing systems", Ph.D. dissertation, University of Stanford, Doc. 40-10-1, Stanford Comput. Center, Stanford, Calif. 1967.
29. Nielsen, N.R. "The simulation of time sharing systems", Comm. of ACM, July 1967, vol. 10, no. 7, pp. 397-412.
30. O'Neill, L.A. "Interactive tolerance analysis with graphic display", Spring Joint Computer Conference, 1969, p. 207.
31. Orlicky, J.A. "Computer selection", Computers and Automation, Sept 1968, pp. 44-49.
32. Rosa, J. "Command and control display technology since 1962", Second Congress on Information System Sciences, 1965, pp. 411-414.
33. Shackel, B. "Man-computer interaction - the contribution of the human sciences", IEEE Trans. Man-Machine Systems, Dec 1969, vol. MMS-10, part II, pp. 149-163.
34. Shackel, B. and P. Shipley. "Man-computer interaction: a review of ergonomics literature and related research", EMI Electronics Report DMP 3472, Feltham, Middlesex, England, 1970.
35. Sackman, H. "Computer, systems science and evolving society", John Wiley and Sons Inc., New York, 1967.
36. Siegel, A.I. and W. Miehle. "The DEI technique for evaluating equipment systems from the information transfer point of view", Human Factors, 1964, no. 6, pp. 279-286.
37. Summers, J.K. and E. Bennett. "AESOP - a final report: a prototype on-line interactive information control system", Third Congress on Information System Science and Technology, Thompson Book Company, Washington, D.C., 1967, pp. 69-86.

APPENDIX A

DESCRIPTION OF THE SIMULATOR MONITOR

The monitor provides interfacing program logic necessary for an on-line user at GRID [20] to communicate with and control the time-sharing simulator used in the experiment. With appropriate modifications, it could be used equally well with other general purpose simulators on the GRID system.

The program is described in general terms with reference to sections of the FORTRAN source listing which is included. Only those subroutines called from the monitor and not included in the GRID support package are described. Their construction was straight forward and therefore it was not considered necessary to include their source listings here also.

The monitor, called subroutine GRAFIC, receives control from MTS [26] after the GRID system has been attached to the /360 system. Statements in the source listing up to the block identification primarily define the display blocks necessary. Nine main options are available to a user (described in Appendix B) and are displayed to him prior to statement 1000. Suboptions from most of these are selectable from within each of the statement groups: 1000, 2000, ... , 9000.

Part of Nielsen's MAIN program was included within GRAFIC. This portion immediately succeeds statement 6000 and hands control to the simulator for a preselected time interval contained in variable TSTOP. On each return from the simulator (from TIMER), a user has the same nine

options again accessible, beginning at statement 20.

Nielsen's model was designed to pass variables between the original thirty-one subroutines within by way of a COMMON area of $24855 * 4$ words. GRAFIC has access to these simulator variables through array VAR.

Those subroutines called from GRAFIC and explicitly used for this routine are described:

SET: used to set a display block as being light pen non-detectable.

LENG: used to interpret the field lengths of digits returned within character strings from GRID.

FIND: searches a catalog file containing indices of all variable names used within Nielsen's model. An index may then be used as a pointer to variables in array VAR.

PLACE: adjusts character arrays reflecting hardware configuration changes. It is called each time a device count changes.

DSPYV: A modification was made to Nielsen's model (in TIMER) in order that selected variables could be sampled every TSAMP period. Time and magnitude changes are collected in array STAT as the simulator runs. End around overlay was used for the cases when more than 100 changes occurred during a run period. When the display option is directed by a user, DSPYV may be called to arrange the time series collected in STAT into an acceptable display graph for viewing at GRID.

EDGE: merely provides scale markers around boundary containing time series plots created by DSPYV.

IRAN7 and IRANL: random number generators used by Nielsen's model.

Must be set for each iteration of validation experiment to ensure identical initial settings.

RECONF: called each time a new configuration of hardware has been arranged. Essentially, it sets up a data file (on logical unit 9) to be read by subroutine RINITL.

RINITL: Nielsen's routines INITL, INIT2 and CONST are combined and shortened into RINITL. This routine initializes COMMON area on each iteration of experiment. Immediately preceding a call to this code, part of VAR is initialized by reading a standard file into VAR.

TIMER: this is Nielsen's control routine for the simulator. It was modified to return control every TSTOP period.

DONCAL and ITRSUM: Nielsen routines called from original MAIN and therefore necessary in GRAFIC. ITRSUM merely reduces statistics collected.

EVAL: contains the essential code from Nielsen's routines of FINSUM and FIN2. It reduces statistics collected and calculates Worthiness indices as described in Chapter IV.

In order to use the monitor as currently implemented, it is important that logical devices be identified prior to object time. Such units are referenced from GRAFIC, Nielsen's model and the supporting routines described above. Currently, the devices are assigned as follows:

DSRN 0: a file containing the minimum hardware configuration data in card image form.

DSRN 1: a file assigned to Nielsen's statistics gathering routines.

DSRN 2: a file assigned to Nielsen's statistics gathering routines.

DSRN 3: catalog file containing indices of variables in COMMON.

DSRN 4: a file assigned to Nielsen's statistics gathering routines.

DSRN 5: contains portion of initialization data read by RINITL.

DSRN 6: output information directed from the simulator (RINITL and EVAL).

- DSRN 7: file containing standard initialized image of COMMON area.
- DSRN 8: dedicated to /360 for directing communication with GRID device.
- DSRN 9: file built by RECONF, from information in logical unit 0, and read at initialization time by RINITL.
- DSRN 10: currently assigned to an MTS 'SINK', such as in IBM 2741 or a line printer (if running a batch job). Used to record pertinent events.

The source listing of GAAFIC begins on the following page.


```

SUBROUTINE GRAFIC
LOGICAL*1 REL/.TRUE./,NOTREL/.FALSE./,BLINK/.TRUE./,
1NOBLK/.FALSE./,LARGE/.TRUE./,SMALL/.FALSE./,
2NODASH/.FALSE./,NODET/.FALSE./,JUMP,JMP,JMP1,EXIT
LOGICAL*1 ERROR,P(20),END(10),ST(12),CALL/.TRUE./,NOCALL/.FALSE./
INTEGER*2 ARR14(20)/'SAMPLE RATE CURRENTLY: 100.0 MS '/'
INTEGER*2 ARR16(20)/'STOP RATE CURRENTLY: 5.50 SECS '/'
INTEGER*2 ARR18(20)/'CURRENT JOB SEQUENCE IS: '/'
INTEGER*2 ARR24(23)/'VARIABLE WAS NOW
1 '/'
INTEGER*2 ARR26A(20)/' 1 CENTRAL PROCESSING UNITS '/'
INTEGER*2 ARR26B(10)/' 100 CORE PAGES '/'
INTEGER*2 ARR26C(15)/' 1 PAGING DRUMS '/'
INTEGER*2 ARR26D(15)/' 1 PAGING DISKS '/'
INTEGER*2 ARR26E(20)/' 3 INPUT / OUTPUT CHANNELS '/'
INTEGER*2 ARR26F(15)/' 1 MAGNETIC TAPE UNITS '/'
INTEGER*2 ARR26G(15)/' 1 LINE PRINTERS '/'
INTEGER*2 ARR26H(15)/' 1 CARD READERS '/'
INTEGER*2 ARR26I(15)/' 1 RANDOM ACCESS UNITS '/'
INTEGER*2 ARR26J(15)/' 0 OTHER DEVICES '/'
INTEGER*2 ARR30(21)/'VARIABLE STATIONARY AT '/'
INTEGER*2 ARR38A(9)/'VARIABLE '/'
INTEGER*2 ARR38B(8)/'T1: SEC '/'
INTEGER*2 ARR38C(20)/'CURRENT VALUE OF IS: '/'
INTEGER*2 ARR38D(6)/'MAX: '/'
INTEGER*2 ARR38E(6)/' N: '/'
INTEGER*2 ARR38F(8)/'T2: SEC '/'
INTEGER*2 ARR50(20)/'SIMULATED TIME: 0.0 SECONDS '/'
INTEGER*2 STRING(10),NN(6),BL/' ',X(201),Y(201),ARRAY(5),RR(4)
INTEGER BLK(8),TYPE,BK,STAT(20,100), BLANK/' ',FIND
INTEGER N(3),TSAMP,TSTOP,TIME/0/,TSET1,TSET2
INTEGER M(3),TLAST,T1,T2,FRONT(10),SPOT(10)
REAL*8 MNAME(10),RNAME,B/' '/'
EQUIVALENCE (P,STRING,RNAME,RR),(N,NN,ST),(ARR38C(10),ARRAY(1))
EQUIVALENCE (VAR(92),TIME),(VAR(41),TSAMP),(VAR(42),TSTOP)
DATA NCPU/1/,NCORE/100/,NDRUM/1/,NDISK/1/,NCHN/3/,NTAPE/1/,
1NPRINT/1/,NREAD/1/,N2314/1/,NOTHR/0/
COMMON /AREA1/ STAT,FRONT,SPOT,NVMON,X,Y,JB(4),END
INTEGER VAR(24855)
COMMON VAR

CALL BLOCK(1)
CALL TEXT(NOTREL,20,10,20,'INITIALIZE SIMULATOR',LARGE,NOBLK)
CALL ENDBLK

CALL BLOCK(2)
CALL TEXT(NOTREL,20,10,26,'CHANGE SIMULATOR VARIABLES',LARGE,
1NOBLK)
CALL ENDBLK

CALL BLOCK(3)
CALL TEXT(NOTREL,20,10,25,'CHANGE CONTROL PARAMETERS',LARGE,
1NOBLK)
CALL ENDBLK

```



```
C      CALL BLOCK(4)
      CALL TEXT(NOTREL,20,10,36,'PERTURB OR VIEW SYSTEM CONFIGURATION',
1LARGE,NOBLK)
      CALL ENDBLK

C      CALL BLOCK(5)
      CALL TEXT(NOTREL,20,10,7,'DISPLAY',LARGE,NOBLK)
      CALL ENDBLK

C      CALL BLOCK(6)
      CALL TEXT(NOTREL,20,10,20,'RUN SIMULATED SYSTEM',LARGE,NOBLK)
      CALL ENDBLK

C      CALL BLOCK(7)
      CALL TEXT(NOTREL,20,10,15,'EVALUATE SYSTEM',LARGE,NOBLK)
      CALL ENDBLK

C      CALL BLOCK(8)
      CALL TEXT(NOTREL,20,10,19,'PREEMPT CURRENT JOB',LARGE,NOBLK)
      CALL ENDBLK

C      CALL BLOCK(9)
      CALL TEXT(NOTREL,20,10,14,'STOP SIMULATOR',LARGE,NOBLK)
      CALL ENDBLK

C      CALL BLOCK(10)
      CALL TEXT(NOTREL,400,600,24,'INITIALIZE CONFIGURATION',
1LARGE,NOBLK)
      CALL ENDBLK

C      CALL BLOCK(12)
      CALL TEXT(NOTREL,200,500,44,'TYPE IN VARIABLES FOR MONITORING + IN
1DEX MOD',SMALL,NOBLK)
      CALL ENDBLK

C      CALL BLOCK(20)
      CALL TEXT(NOTREL,500,150,29,'TYPE IN ANY ADJUSTMENT WANTED',
1LARGE,NOBLK)
      CALL ENDBLK

C      CALL BLOCK(22)
      CALL TEXT(NOTREL, 0, 0,44,'TYPE IN VARIABLE + NEW VALUE + ANY IN
1DEX MOD',SMALL,NOBLK)
      CALL ENDBLK

C      CALL BLOCK(26)
      CALL TEXT(NOTREL,0,0,37,'CURRENT SYSTEM RESOURCE CONFIGURATION',
1LARGE,NOBLK)
      CALL ENDBLK

C      CALL BLOCK(28)
      CALL TEXT(NOTREL,0,0,28,'TYPE IN VARIABLE FOR DISPLAY',
1LARGE,NOBLK)
      CALL ENDBLK
```



```

C      CALL BLOCK(32)
      CALL TEXT(NOTREL,20,10,28,'DISPLAY A MONITORED VARIABLE',
1LARGE,NOBLK)
      CALL ENDBLK

C      CALL BLOCK(34)
      CALL TEXT(NOTREL,20,10,25,'DISPLAY VALUE OF VARIABLE',LARGE,NOBLK)
      CALL ENDBLK

C      CALL BLOCK(36)
      CALL TEXT(NOTREL,20,10,25,'DISPLAY LATEST STATISTICS',LARGE,NOBLK)
      CALL ENDBLK

C      K=10
      CALL BLOCK(38)
      CALL MOVE(NOTREL,110,312)
      CALL VECTOR(REL,800,0,NOBLK,NODASH)
      CALL VECTOR(REL,0,400,NOBLK,NODASH)
      CALL VECTOR(REL,-800,0,NOBLK,NODASH)
      CALL VECTOR(REL,0,-400,NOBLK,NODASH)
      CALL EDGE(110,312,100,K,0,-K)
      CALL EDGE(910,312,-K,50,K,0)
      CALL EDGE(910,712,-100,-K,0,K)
      CALL EDGE(110,712,K,-50,-K,0)
      CALL ENDBLK

C      CALL BLOCK(40)
      CALL TEXT(NOTREL,320,510,34,'DISTRIBUTION OF INTERACTION FACTORS',
1LARGE,NOBLK)
      CALL TEXT(NOTREL,320,460,8,'JOB DATA',LARGE,NOBLK)
      CALL TEXT(NOTREL,320,410,8,'CPU DATA',LARGE,NOBLK)
      CALL TEXT(NOTREL,320,360,11,'PAGING DATA',LARGE,NOBLK)
      CALL TEXT(NOTREL,320,310,10,'QUEUE DATA',LARGE,NOBLK)
      CALL TEXT(NOTREL,320,260,11,'DEVICE DATA',LARGE,NOBLK)
      CALL ENDBLK

C      CALL BLOCK(75)
      CALL TEXT(NOTREL, 0, 0,24,'SYNTAX ERROR - TRY AGAIN',
1LARGE,BLINK)
      CALL ENDBLK

C      CALL BLOCK(80)
      CALL TEXT(NOTREL, 0, 0,19,'ERROR IN CONVERSION',LARGE,BLINK)
      CALL ENDBLK

C      CALL BLOCK(85)
      CALL TEXT(NOTREL,650,850,24,'SIMULATION RUN COMPLETED',LARGE,REL)
      CALL ENDBLK

C      DO 5 BK=100,350,50
      CALL BLOCK(BK)
      CALL MOVE(NOTREL,0,0)
      CALL VECTOR(REL,0,20,NOBLK,NODASH)
      CALL VECTOR(REL,-20,0,NOBLK,NODASH)

```


CALL VECTOR(REL,0,-20,NOBLK,NODASH)

CALL VECTOR(REL,20,0,NOBLK,NODASH)

5 CALL ENDBLK

C CALL BLOCK(101)

CALL TEXT(NOTREL,700,850,17,'CONTROL INTERRUPT',SMALL,BLINK)

CALL ENDBLK

C CALL BLOCK(102)

CALL TEXT(NOTREL,0,0,31,'NEILSON SIMULATOR/TSS VERSION 1',
1LARGE,NOBLK)

CALL ENDBLK

C CALL BLOCK(260)

CALL TEXT(NOTREL,100,650,42,'CURRENT RESOURCE CONFIGURATION NOW INC
1LUDES',SMALL,NOBLK)

CALL ENDBLK

C CALL BLOCK(261)

CALL TEXT(NOTREL,370,610,40,ARR26A,SMALL,NOBLK)

CALL ENDBLK

C CALL BLOCK(262)

CALL TEXT(NOTREL,370,570,20,ARR26B,SMALL,NOBLK)

CALL ENDBLK

C CALL BLOCK(263)

CALL TEXT(NOTREL,370,530,25,ARR26C,SMALL,NOBLK)

CALL ENDBLK

C CALL BLOCK(264)

CALL TEXT(NOTREL,370,490,25,ARR26D,SMALL,NOBLK)

CALL ENDBLK

C CALL BLOCK(265)

CALL TEXT(NOTREL,370,450,35,ARR26E,SMALL,NOBLK)

CALL ENDBLK

C CALL BLOCK(266)

CALL TEXT(NOTREL,370,410,30,ARR26F,SMALL,NOBLK)

CALL ENDBLK

C CALL BLOCK(267)

CALL TEXT(NOTREL,370,370,25,ARR26G,SMALL,NOBLK)

CALL ENDBLK

C CALL BLOCK(268)

CALL TEXT(NOTREL,370,330,25,ARR26H,SMALL,NOBLK)

CALL ENDBLK

C CALL BLOCK(269)

CALL TEXT(NOTREL,370,290,29,ARR26I,SMALL,NOBLK)

CALL ENDBLK

C CALL BLOCK(270)


```

CALL TEXT(NOTREL,370,250,30,ARR26J,SMALL,NOBLK)
CALL ENDBLK
CALL SET(10,1,0,0)
CALL SET(12,1,0,0)
CALL SET(20,1,0,0)
CALL SET(22,1,300,200)
CALL SET(26,1,250,800)
CALL SET(28,1,350,350)
CALL SET(32,1,400,600)
CALL SET(34,1,400,500)
CALL SET(36,1,400,400)
CALL SET(38,1,0,0)
CALL SET(75,1,375,750)
CALL SET(80,1,400,700)
CALL SET(85,1,0,0)
CALL SET(101,1,0,0)
CALL SET(102,1,250,900)

```

C

C

```

DISPLAY INITIAL MONITOR AND SETUP LP BLOCKS

```

```

K=700

```

```

J=600

```

```

DO 10 BK=1,9

```

```

CALL DISPLY(100,BK,300,K)

```

```

CALL DISPLY(200,BK,350,J)

```

```

IF(ID.LE.8) CALL DISPLY(300,BK,300,J)

```

```

CALL DISPLY(BK,1,300,K)

```

```

CALL LP(BK,1,NODET)

```

```

J=J-40

```

```

10 K=K-50

```

```

CALL DISPLY(200,10,350,J)

```

```

CALL ERSBK(200)

```

```

CALL ERSBK(300)

```

C

```

CALL DISPLY(150,1,380,590)

```

```

CALL ERSBK(150)

```

C

```

CALL DISPLY(350,1,300,500)

```

```

CALL DISPLY(350,2,300,400)

```

```

CALL ERSBK(350)

```

C

```

CALL DISPLY(250,1,400,600)

```

```

CALL DISPLY(250,2,400,500)

```

```

CALL DISPLY(250,3,400,400)

```

```

CALL ERSBK(250)

```

```

CALL RSTBK(102)

```

C

C

```

SET FLAG FOR JOB PREEMPTION

```

```

VAR(149)=0

```

C

```

TSET1=1000

```

```

TSET2=55000

```

C

C

```

SET INITIAL JOB STREAM

```

```

JB(1)=10

```

```

JB(2)=4

```


JB(3)=10

JB(4)=4

GO TO 35

```

C
C BLOCK IDENTIFICATION
C 1  INITIALIZE SIMULATOR
C 2  CHANGE SIMULATOR VARIABLES
C 3  CHANGE CONTROL PARAMETERS
C 4  PERTURB OR VIEW SYSTEM CONFIGURATION
C 5  DISPLAY
C 6  RUN SIMULATED SYSTEM
C 7  EVALUATE SYSTEM
C 8  PREEMPT CURRENT JOB
C 9  STOP SIMULATOR
C 10 INITIALIZE COMMON VARIABLES BY EITHER:
C      READ DATA
C      READ INITIALIZED SEQUENTIAL FILE
C 12 TYPE IN VARIABLES TO BE MONITORED
C 14 SAMPLE RATE CURRENTLY
C 16 STOP RATE CURRENTLY
C 18 CURRENT JOB SEQUENCE IS:
C 20 TYPE IN ANY ADJUSTMENT WANTED
C 22 TYPE IN VARIABLE + NEW VALUE + ANY INDEX MOD
C 24 VARIABLE WAS NOW
C 26 CURRENT SYSTEM RESOURCE CONFIGURATION (A-J)
C 28 TYPE IN VARIABLE NAME FOR DISPLAY
C 30 THIS MONITORED VARIABLE DID NOT VARY
C 32 DISPLAY A MONITORED VARIABLE
C 34 DISPLAY VALUE OF VARIABLE
C 36 DISPLAY LATEST STATISTICS
C 37 MONITOR VARIABLE BLOCK
C 38 BOUNDARY FOR MONITORED VARIABLES
C 40 DISTRIBUTION OF INTERACTION FACTORS
C      JOB DATA
C      CPU DATA
C      PAGING DATA
C      QUEUE DATA
C      DEVICE DATA
C 50 SIMULATED TIME:      SECONDS
C 75 SYNTAX ERROR - TRY AGAIN
C 80 CONVERSION ERROR
C 85 SIMULATION RUN COMPLETED
C 100 QUADS FOR LP'S
C 101 CONTROL INTERRUPT
C 102 NEILSON SIMULATOR/TSS VERSION 1
C 150 QUADS FOR LP'S
C 200 QUADS FOR LP'S
C 250 QUADS FOR LP'S
C 300 QUADS FOR LP'S
C
C ***** RETURN FROM TIMER HERE FOR CONTROL STOP *****
333 CALL RSTBK(101)
20  IF(TIME.EQ.TLAST) GO TO 25
    TLAST=TIME
C

```



```

IF(TIME.GT.0) CALL DELBLK(50)
CALL FLTCHR(TIME/10000.0,ARR50(9),8,4)
CALL BLOCK(50)
CALL TEXT(NOTREL,300,800,40,ARR50,SMALL,NOBLK)
CALL ENDBLK
CALL DISPLY(50,1,0,0)
CALL LP(50,1,NODET)

```

```

C
25 IF(JMP) GO TO 35
   CALL RSTBK(50)
   CALL RSTBK(100)
   CALL ERSBK(102)
   DO 30 BK=1,9
30  CALL RSTBK(BK)

```

```

C
35 JMP=.FALSE.
   CALL TRANMT
   CALL ERSBK(85)
   CALL ERSBK(101)
   CALL DLPEN(1,IX,IY,TYPE,ID,BLK,&45)
   CALL DEND(1,&35)
40  CALL RSTBK(75)
   GO TO 35
45  CALL ERSBK(75)
   CALL ERSBK(101)
   DO 50 BK=1,9
   IF(BK.NE.ID) CALL ERSBK(BK)
50  CONTINUE
   IF(ID.LT.6) CALL MOVEBK(ID,1,0,950)
   CALL ERSBK(100)

```

```

C
1000 GO TO (1000,2000,3000,4000,5000,6000,7000,8000,9000),ID
      CALL RSTBK(10)
      CALL ERSBK(75)
      CALL RSTBK(150)

```

```

C
55  CALL TRANMT
      CALL ERSBK(10)
      CALL ERSBK(150)
      CALL DLPEN(1,IX,IY,TYPE,ID,BLK,&65)
      CALL DEND(1,&67)
      CALL RSTBK(75)
      GO TO 1000
65  REWIND 5
      REWIND 7
      REWIND 9
      IF(TIME.GT.0) CALL DELBLK(50)
      NVMON=0
      READ(7) VAR

```

```

C
      CALL RINITL
      III=IRANL(1)
      JJJ=IRAN7(1)
      TSTOP=55000
      TSAMP=1000

```



```

C
C ANY VARIABLES TO BE MONITORED?
67 CALL RSTBK(12)
   CALL ERSBK(75)
   VAR(51)=1000000
C
70 CALL TRANMT
   CALL DALPHA(1,IX,IY,NUM,10,STRING,&75)
   CALL DEND(1,&90)
73 CALL RSTBK(75)
   GO TO 70
C
C CHECK IF STRING FORMAT IS CORRECT
75 NUM2=NUM*2
   CALL ERSBK(75)
   CALL ERSBK(80)
   K=LENG(1,NUM2,P,CALL)
   N(1)=BLANK
   N(2)=BLANK
   DO 80 I=1,K
80 ST(I)=P(I)
C
   INDEX=FIND(N)
C
   IF(INDEX.EQ.0) GO TO 73
   NVMON=NVMON+1
   SPOT(NVMON)=INDEX
C CHECK FOR AN INDEX MODIFIER
   IF(K+1.GE.NUM2) GO TO 85
   L=K+2
   K=LENG(L,NUM2,P,NOCALL)
   SPOT(NVMON)=SPOT(NVMON)+INTCVT(P(L),K,ERROR)-1
   IF(.NOT. ERROR) GO TO 85
   CALL RSTBK(80)
   GO TO 70
85 FRONT(NVMON)=1
   END(NVMON)=.FALSE.
   NV=NVMON*2
   STAT(NV-1,1)=VAR(INDEX)
   STAT(NV,1)=TIME
   MNAME(NVMON)=RNAME
   CALL ERSBK(75)
   GO TO 70
C
C ANY CHANGE TO CURRENT JOB SEQUENCE DESIRED?
90 CALL ERSBK(12)
   CALL ERSBK(75)
   CALL ERSBK(80)
C PUT JOB SEQUENCE NUMB. IN ARR18
   CALL INTCHR(JB(1),ARR18(14),4,2)
   CALL INTCHR(JB(2),ARR18(16),4,2)
   CALL INTCHR(JB(3),ARR18(18),4,2)
   CALL INTCHR(JB(4),ARR18(20),4,2)
   CALL BLOCK(18)
   CALL TEXT(NOTREL,0,0,40,ARR18,LARGE,NOBLK)

```



```

CALL ENDBLK
CALL DISPLY(18,1,200,500)
CALL RSTBK(20)
C
95  CALL TRANMT
    CALL DELBLK(18)
    CALL DALPHA(1,IX,IY,NUM,10,STRING,&100)
    CALL DEND(1,&110)
    CALL RSTBK(75)
    GO TO 95
97  CALL RSTBK(80)
    GO TO 95
100 NUM2=NUM*2
    K=LENG(1,NUM2,P,CALL)
    JB(1)=INTCVT(P,K,ERROR)
    IF(ERROR) GO TO 97
    IP=1
    DO 105 I=2,4
    IP=IP+K+1
    K=LENG(IP,NUM2,P,NOCALL)
    JB(I)=INTCVT(P(IP),K,ERROR)
    IF(ERROR) GO TO 97
105 CONTINUE
    GO TO 90
C
110 CALL ERSBK(75)
    CALL ERSBK(80)
    CALL ERSBK(20)
C
C  DISPLAY SAMPLE RATE IN ARRAY 14
    IF(TSAMP.EQ.TSET1) GO TO 112
    TSET1=TSAMP
    CALL FLTCHR(TSAMP/10.0,ARR14(12),8,1)
112 CALL BLOCK(14)
    CALL TEXT(NOTREL, 0, 0,40,ARR14,LARGE,NOBLK)
    CALL ENDBLK
    CALL DISPLY(14,1,250,500)
    CALL LP(14,1,NODET)
    CALL RSTBK(20)
C
115 CALL TRANMT
    CALL DALPHA(1,IX,IY,NUM,10,STRING,&125)
    CALL DEND(1,&130)
    CALL RSTBK(75)
    GO TO 115
120 CALL RSTBK(80)
    GO TO 115
125 K=LENG(1,NUM*2,P,CALL)
    XTIME=CHRFLT(P,K,ERROR)
    IF(ERROR) GO TO 120
    TSAMP=XTIME*10
C
    CALL PLACE(ARR14,20,STRING,NUM,14,250,500,13,LARGE)
130 CALL DELBLK(14)
    CALL ERSBK(75)

```



```

C
C      PUT STOP RATE IN ARR16
      IF(TSTOP.EQ.TSET2) GO TO 132
      TSET2=TSTOP
      CALL FLTCHR(TSTOP/10000.0,ARR16(11),8,4)
132   CALL BLOCK(16)
      CALL TEXT(NOTREL, 0, 0,40,ARR16,LARGE,NOBLK)
      CALL ENDBLK
      CALL DISPLY(16,1,250,500)
      CALL LP(16,1,NODET)

C
135   CALL TRANMT
      CALL DALPHA(1,IX,IY,NUM,10,STRING,&145)
      CALL DEND(1,&150)
      CALL RSTBK(75)
      GO TO 135
140   CALL RSTBK(80)
      GO TO 135
145   K=LENG(1,NUM*2,P,CALL)
      XTIME=CHRFLT(P,K,ERROR)
      IF(ERROR) GO TO 140
      TSTOP=XTIME*10000
      CALL PLACE(ARR16,20,STRING,NUM,16,250,500,12,LARGE)
150   CALL DELBLK(16)
      CALL ERSBK(20)
      CALL ERSBK(75)
      CALL MOVEBK(1,1,300,700)
      GO TO 20

C      ***** 2000 *****
2000  CALL RSTBK(22)
      JUMP=.FALSE.
155   CALL TRANMT
      IF(JUMP) CALL DELBLK(24)
      CALL DALPHA(1,IX,IY,NUM,10,STRING,&165)
      CALL DEND(1,&195)
160   CALL RSTBK(75)
      GO TO 155
165   CALL ERSBK(75)
      CALL ERSBK(80)
      NUM2=NUM*2
      K=LENG(1,NUM2,P,CALL)

C
C      DECODE VARIABLE INTO ARRAY N
      N(1)=BLANK
      N(2)=BLANK
      N(3)=BLANK
      DO 170 I=1,K
170   ST(I)=P(I)
      INDEX=FIND(N)
      IF(INDEX.EQ.0) GO TO 160

C
C      CHECK WHAT NEW VALUE SHOULD BECOME
      IF(K+1.GE.NUM2) GO TO 160
175   L=K+2
      K=LENG(L,NUM2,P,NOCALL)

```



```

NEWVAL=INTCVT(P(L),K,ERROR)
IF(ERROR) GO TO 180
C
C CHECK FOR ANY INDEX MODIFICATION
IF(L+K.GE.NUM2) GO TO 185
L=L+K+1
K=LENG(L,NUM2,P,NOCALL)
INDEX=INDEX+INTCVT(P(L),K,ERROR)-1
IF(.NOT.ERROR) GO TO 185
180 CALL RSTBK(80)
GO TO 155
C
C PUT VARIABLE WAS XXX NOW YYY IN ARR24
185 CALL INTCHR(VAR(INDEX),ARR24(13),4,8)
CALL INTCHR(NEWVAL,ARR24(20),4,8)
VAR(INDEX)=NEWVAL
DO 190 I=1,5
190 ARR24(I+5)=NN(I)
JUMP=.TRUE.
CALL BLOCK(24)
CALL TEXT(NOTREL,0,0,46,ARR24,LARGE,NOBLK)
CALL ENDBLK
CALL DISPLY(24,1,250,500)
GO TO 155
195 CALL ERSBK(22)
CALL ERSBK(75)
CALL ERSBK(80)
CALL MOVEBK(2,1,300,650)
GO TO 20
C
C ***** 3000 *****
3000 CALL RSTBK(20)
CALL RSTBK(350)
C
C DISPLAY SAMPLE RATE IN ARRAY ARR14
CALL BLOCK(14)
CALL TEXT(NOTREL,0,0,40,ARR14,LARGE,NOBLK)
CALL ENDBLK
CALL DISPLY(14,1,320,510)
CALL LP(14,1,NODET)
C
C DISPLAY STOP RATE IN ARRAY ARR16
CALL BLOCK(16)
CALL TEXT(NOTREL,0,0,40,ARR16,LARGE,NOBLK)
CALL ENDBLK
CALL DISPLY(16,1,320,410)
CALL LP(16,1,NODET)
C
210 CALL TRANMT
CALL DLPEN(1,IX,IY,TYPE,ID,BLK,&220)
CALL DEND(1,&245)
215 CALL RSTBK(75)
GO TO 210
220 CALL DALPHA(2,IX,IY,NUM,10,STRING,&225)
GO TO 215
225 CALL ERSBK(75)

```



```

CALL ERSBK(80)
K=LENG(1,NUM*2,P,CALL)
XTIME=CHRFLT(P,K,ERROR)
IF(.NOT.ERROR) GO TO 230
CALL RSTBK(80)
GO TO 210
230 GO TO (235,240) ,ID
235 TSAMP=XTIME*10
TSET1=TSAMP
CALL PLACE(ARR14,20,STRING,NUM,14,320,510,13,LARGE)
GO TO 210
240 TSTOP=XTIME*10000
TSET2=TSTOP
CALL PLACE(ARR16,20,STRING,NUM,16,320,410,12,LARGE)
GO TO 210
245 CALL DELBLK(14)
CALL DELBLK(16)
CALL ERSBK(20)
CALL ERSBK(350)
CALL MOVEBK(3,1,300,600)
GO TO 20
C ***** 4000 *****
4000 JMP1=.TRUE.
250 JUMP=.FALSE.
CALL RSTBK(20)
CALL RSTBK(26)
IF(TIME.GT.0) CALL ERSBK(50)
CALL ERSBK(102)
CALL RSTBK(200)
DO 350 BK=261,270
CALL DISPLY(BK,1,0,0)
350 CALL LP(BK,1,NODET)
C
355 CALL TRANMT
CALL DLPEN(1,IX,IY,TYPE,ID,BLK,&360)
CALL DEND(1,&359)
CALL RSTBK(75)
GO TO 350
356 CALL RSTBK(75)
GO TO 355
359 JUMP=.TRUE.
GO TO 365
360 CALL DALPHA(2,IX,IY,NUM,10,STRING,&361)
GO TO 356
361 K=LENG(1,NUM*2,P,CALL)
NEWVAL=INTCVT(P,K,ERROR)
IF(.NOT.ERROR) GO TO 365
CALL RSTBK(80)
GO TO 355
365 DO 370 BK=261,270
370 CALL FREEBK(BK)
CALL ERSBK(20)
CALL ERSBK(26)
CALL ERSBK(75)
CALL ERSBK(80)

```



```

CALL ERSBK(200)
IF(JUMP) GO TO 390
CALL DISPLY(260,1,0,0)
GO TO (371,372,373,374,375,376,377,378,379,380),ID
371 NCPU=NEWVAL
CALL PLACE(ARR26A,20,STRING,NUM,261,370,610,2,SMALL)
GO TO 385
372 NCORE=NEWVAL
CALL PLACE(ARR26B,10,STRING,NUM,262,370,570,2,SMALL)
GO TO 385
373 NDRUM=NEWVAL
CALL PLACE(ARR26C,15,STRING,NUM,263,370,530,2,SMALL)
GO TO 385
374 NDISK=NEWVAL
CALL PLACE(ARR26D,15,STRING,NUM,264,370,490,2,SMALL)
GO TO 385
375 NCHN=NEWVAL
CALL PLACE(ARR26E,20,STRING,NUM,265,370,450,2,SMALL)
GO TO 385
376 NTAPE=NEWVAL
CALL PLACE(ARR26F,15,STRING,NUM,266,370,410,2,SMALL)
GO TO 385
377 NPRINT=NEWVAL
CALL PLACE(ARR26G,15,STRING,NUM,267,370,370,2,SMALL)
GO TO 385
378 NREAD=NEWVAL
CALL PLACE(ARR26H,15,STRING,NUM,268,370,330,2,SMALL)
GO TO 385
379 N2314=NEWVAL
CALL PLACE(ARR26I,15,STRING,NUM,269,370,290,2,SMALL)
GO TO 385
380 NOTHR=NEWVAL
CALL PLACE(ARR26J,15,STRING,NUM,270,370,250,2,SMALL)
385 CALL MOVEBK(ID+260,1,210,ID*40)
C
CALL TRANMT
JMP1=.FALSE.
CALL FREEBK(ID+260)
CALL FREEBK(260)
GO TO 250
390 CALL MOVEBK(4,1,300,550)
IF(JMP1) GO TO 20
CALL RECONF(NCPU,NCORE,NDRUM,NDISK,NTAPE,NPRINT,NREAD,N2314)
GO TO 20
C
***** 5000 *****
5000 CALL RSTBK(32)
CALL RSTBK(34)
CALL RSTBK(36)
CALL RSTBK(250)
EXIT=.FALSE.
C
400 CALL TRANMT
CALL ERSBK(32)
CALL ERSBK(34)
CALL ERSBK(36)

```



```

CALL ERSBK(75)
CALL ERSBK(250)
CALL DLPEN(1,IX,IY,TYPE,ID,BLK,&410)
CALL DEND(1,&406)
405 CALL RSTBK(75)
GO TO 5000
406 CALL MOVEBK(5,1,300,500)
GO TO 20
410 CALL ERSBK(75)
CALL ERSBK(80)
GO TO (415,460,500), ID
C
C MONITORED VARIABLE TO BE DISPLAYED
415 IF(NVMON.EQ.0) GO TO 405
RNAME=B
CALL DALPHA(2,IX,IY,NUM,10,STRING,&420)
M1=1
M2=NVMON
GO TO 430
420 DO 425 MARK=1,NVMON
IF(RNAME.EQ.MNAME(MARK)) GO TO 429
425 CONTINUE
GO TO 405
C
C MARK POINTS TO MONITORED VARIABLE FOR DISPLAY
429 M1=MARK
M2=MARK
430 DO 455 MARK=M1,M2
RNAME=MNAME(MARK)
IF(FRONT(MARK).GT.1.OR.END(MARK)) GO TO 435
DO 431 I=6,9
431 ARR30(I)=RR(I-5)
CALL INTCHR(VAR(SPOT(MARK)),ARR30(18),4,8)
CALL BLOCK(30)
CALL TEXT(NOTREL,175,550,42,ARR30,LARGE,NOBLK)
CALL ENDBLK
CALL DISPLY(30,1,0,0)
C
CALL TRANMT
CALL DELBLK(30)
GO TO 455
435 CALL DSPYV(MARK,T1,T2,MAX,NPTS)
C
C PLACE VARIABLE NAME IN ARR38A
DO 440 I=6,9
440 ARR38A(I)=RR(I-5)
CALL RSTBK(38)
C
C PLACE LOWER AND UPPER TIME BOUNDS IN ARR38B
CALL FLTCHR(T1/10000.0,ARR38B(3),8,4)
CALL FLTCHR(T2/10000.0,ARR38F(3),8,4)
C
C PLACE MAXIMUM VALUE OF VARIABLE IN ARR38D
CALL INTCHR(MAX,ARR38D(4),4,6)
C

```



```

C      PLACE NUMBER OF TIMES VARIABLE CHANGED IN ARR38E
      CALL INTCHR((NPTS+1)/2,ARR38E(4),4,6)
      CALL BLOCK(39)
      CALL TEXT(NOTREL,275,438,18,ARR38A,LARGE,NOBLK)
      CALL DLINE(NOTREL,NOTREL,X,Y,NPTS,NOBLK,NODASH)
      CALL TEXT(NOTREL,-110,-20,16,ARR38B,LARGE,NOBLK)
      CALL TEXT(NOTREL,670,-20,16,ARR38F,LARGE,NOBLK)
      CALL TEXT(NOTREL,300,-75,12,ARR38D,LARGE,NOBLK)
      CALL TEXT(NOTREL,300,-115,12,ARR38E,LARGE,NOBLK)
      CALL ENDBLK
      CALL DISPLY(39,1,110,312)

C
      CALL TRANMT
      CALL ERSBK(38)
      CALL DELBLK(39)
455    CONTINUE
      GO TO 5000

C
C      CURRENT VALUE OF A VARIABLE TO BE DISPLAYED
460    CALL RSTBK(28)
C
461    CALL TRANMT
462    CALL DALPHA(1,IX,IY,NUM,10,STRING,&465)
      CALL ERSBK(28)
      CALL DEND(1,&5000)
463    CALL RSTBK(75)
      GO TO 461
465    CALL ERSBK(75)
      CALL ERSBK(80)
      NUM2=NUM*2
      K=LENG(1,NUM2,P,CALL)
      N(1)=BLANK
      N(2)=BLANK
      DO 470 I=1,K
470    ST(I)=P(I)
      INDEX=FINN(N)
      IF(INDEX.EQ.0) GO TO 463

C
C      ADJUST FOR ANY INDEX MOD
      IF(K+1.GE.NUM2) GO TO 480
      L=K+2
      K=LENG(L,NUM2,P,NOCALL)
      NEWVAL=INTCVT(P(L),K,ERROR)
      IF(.NOT.ERROR) GO TO 475
      CALL RSTBK(80)
      GO TO 461
475    INDEX=INDEX+NEWVAL-1
480    DO 481 I=1,5
481    ARRAY(I)=BL
      DO 485 I=1,NUM
485    ARRAY(I)=STRING(I)
C
      CALL INTCHR(VAR(INDEX),ARR38C(17),4,8)
      CALL BLOCK(39)
      CALL TEXT(NOTREL,200,550,40,ARR38C,LARGE,NOBLK)

```



```

CALL ENDBLK
CALL DISPLY(39,1,0,0)
C
CALL TRANMT
CALL DELBLK(39)
GO TO 462
500 CALL RSTBK(300)
CALL DISPLY(40,1,0,0)
CALL TRANMT
C
C ENTER CODE HERE TO DISPLAY STATISTICS WANTED
C
CALL ERSBK(300)
CALL FREEBK(40)
GO TO 5000
C
550 CALL MOVEBK(5,1,300,500)
GO TO 20
C
6000 ***** 6000 *****
CALL TIMER(&333,&666)
VAR(105)=0
REWIND 1
REWIND 2
REWIND 4
VAR(24852)=1
VAR(24853)=1
VAR(24854)=1
IF(VAR(101).NE.0) GO TO 6000
VAR(141)=0
VAR(161)=0
VAR(162)=0
VAR(163)=0
VAR(166)=0
VAR(167)=0
VAR(135)=0
VAR(188)=0
DO 600 I=1,50
VAR(I+436)=0
600 VAR(I+836)=0
DO 605 I=1,4
VAR(I+10054)=TIME
VAR(I+7946)=0
VAR(I+7950)=0
VAR(I+7970)=0
VAR(I+8010)=0
VAR(I+8014)=0
605 VAR(I+8022)=0
VAR(1486+VAR(82))=TIME+VAR(51)
VAR(93)=VAR(82)
VAR(101)=1
CALL DONCAL
610 CALL TIMER(&333,&666)
IF(VAR(101).EQ.0) GO TO 666
VAR(101)=0
CALL ITRSUM

```


GO TO 610

C

C THIS SIMULATION RUN IS FINISHED

666 CALL RSTBK(85)

GO TO 20

C

***** 7000 *****

7000 CALL ITRSUM

FTIME=TIME/10000.0

WRITE(10,790) JB,FTIME,NCPU,NCORE,NDRUM,NDISK,NTAPE,
1NPRINT,NREAD,N2314

790 FORMAT(' SEQUENTIAL JOB STREAM:',4I3,/'SIMULATED RUN TIME:',F10.4,

1' SECS. '/' HARDWARE CONFIGURATION: CPUS ',I3/

126X,'CORE SIZE ',I3/

226X,'PAGING DRUMS',I3/26X,'PAGING DISKS',I3/26X,'TAPE UNITS ',I3/

326X,'PRINTER ',I3/26X,'CARD READERS',I3/26X,'DIR. ACCESS ',I3)

C

C REDUCE SIMULATOR STATISTICS AND CALCULATE WORTHINESS

CALL EVAL(NCPU,NCORE,NDRUM,NDISK,NTAPE,NPRINT,NREAD,N2314)

GO TO 20

C

***** 8000 *****

8000 VAR(149)=1

GO TO 20

C

***** 9000 *****

9000 RETURN

END

APPENDIX B

OPERATING THE EXPERIMENTAL SETTING

As currently implemented at the University of Alberta, the simulator monitor (described in Appendix A) is designed to communicate with the GRID terminal operating under the General Purpose Supervisor [20]. As such, interaction between a user and the /360 system involves the input of a message at the terminal followed by appropriate computation within the /360 and the ultimate presentation of a display at the CRT.

In the experimental setting implemented, only certain input functions were acceptable by the monitor program. These were:

- (a) Light Pen coordinate + SEND function
- (b) Light Pen coordinate + A/N character string + SEND function
- (c) A/N character string + SEND function
- (d) SEND function

For every display presented to a user, the options available for input require one of the above four types. As a standard default input, item (d) is considered as a null instruction to the monitor signifying that no action is required on the current display option(s).

The program executes around the nucleus of nine basic options to a user. A light pen selection of any of these results in various action by the /360. Some of them describe suboptions displayable at the CRT, from which different branches are selectable. The nine primary options are described, as well as their suboptions:

INITIALIZE SIMULATOR: Six suboptions result from this selection:

- (a) simulator may be initialized with current data base configuration.

This action is necessary precluding each configuration test.

- (b) variables may be selected for time-series analysis.
- (c) job stream can be selected.
- (d) sample time may be changed for variables in (b).
- (e) stop time period for simulator may be changed.

CHANGE SIMULATOR VARIABLES: a simulator variable name followed by any index modification and its new desired value may be typed in. The old value, as well as the modified one, appear in a display.

CHANGE CONTROL PARAMETERS: as currently implemented, the sample and stop rates can be adjusted.

PERTURB OR VIEW SYSTEM CONFIGURATION: a display of the current hardware configuration is made. By light pen and A/N, new device resources may be augmented.

DISPLAY: three suboptions are available from this selection:

- (a) display of variable tracked for time-series.
- (b) display of current magnitude of any variable in simulator
(there are 24,855 COMMON variables).
- (c) display latest statistics (not implemented for validation).

RUN SIMULATED SYSTEM: a branch is made to Nielsen's model, and the simulator is run for one stop period.

EVALUATE SYSTEM: the statistics collected are reduced and recorded on logical unit 6. For the validation experiment, Worthiness and cost were also calculated.

PREEMPT CURRENT JOB: used to terminate any simulated job running in the model corresponding to the single user tested in the validation. Also initiates the next job to be run at that simulated terminal.

STOP: a return is made to the control program in MTS.

To add flexibility to the simulator monitor, all incorrectly formatted input messages entered from the GRID terminal are recoverable from. The monitor merely flashes 'ERROR - TRY AGAIN' to the user. Also, to aid those users attempting to study the operation of a simulator by using the monitor, an attempt has been made to display adequate prompting messages for all options. In this way, a user need not have a particularly strong knowledge of all /360 reactions taking place within the monitor.

APPENDIX C

SAMPLE CALCULATION OF WORTHINESS

This appendix is included to illustrate the calculations necessary in the validation experiment to determine Worthiness indices as described in Chapter IV. The magnitudes of the three design parameters of flexibility, response and cost are first calculated from both configuration information and statistics collected during the simulation run. These figures are then used in the Worthiness function, along with Weights and Price functions, to determine a Worthiness index.

The following information assembled from an actual run is used to demonstrate the arithmetic involved:

Statistics collected from Nielsen's model:

Number of interactions: 254

Mean ratio response time: 98.3

Standard Deviation of response: 101.4

Device utilizations: refer to Table C.1

RESPONSE:

$$\begin{aligned} R &= 700 / \text{Mean ratio} + 300 / \text{Standard Deviation} \\ &= 700 / 98.3 + 300 / 101.4 \\ &= 10.08 \end{aligned}$$

COST:

$$C = \sum_{\text{all } j} \text{Unit Cost}_j \cdot (1 + \text{Utilization}_j)$$

[refer Table C.1]

$$= \sum_{\text{all } j} \text{Configuration costs} = 57.87$$

Table C.1. Information necessary to calculate design parameter cost

DEVICE RESOURCE	UNIT COST	UTILIZATION	CONFIGURATION COST
CPU 1	1.0000	48.97	49.970
CORE (166pg)	.0083	not used	1.3788
PG. DRUM 1	.443	0.0	.443
PG. DRUM 2	.443	0.15	.509
PG. DISK 1	.5018	6.95	3.989
TAPE UNIT 1	.0708	0.0	.0708
TAPE UNIT 2	.0708	0.0	.0708
TAPE UNIT 3	.0708	0.0	.0708
TAPE UNIT 4	.0708	0.0	.0708
PRINTER 1	.0677	0.0	.0677
PRINTER 2	.0677	0.0	.0677
PRINTER 3	.0677	0.0	.0677
READ/PUNCH 1	.0506	0.0	.0506
READ/PUNCH 2	.0506	0.0	.0506
DISK 1	.0544	0.0	.0544
DISK 2	.0544	0.0	.0544
DISK 3	.0544	0.76	.0957
DISK 4	.0544	13.49	.788

FLEXIBILITY:

$$F = 0.3 \cdot (\text{no. types devices}) + 0.7 \cdot (\text{no. devices})$$

$$= 0.3 \cdot 9 + 0.7 \cdot 22.53^* = 18.173$$

* Core pages are considered as PAGES / 30 devices

WEIGHTS USED: $W_f = 0.3$

$$W_r = 0.7$$

$$W_c = 1.0$$

PRICE FUNCTIONS USED: $P_f = 30 / F$

$$P_r = 15 / R$$

WORTHINESS FUNCTION:

$$\begin{aligned}
 W &= \frac{F \cdot W_f \cdot P_f + R \cdot W_r \cdot P_r}{C \cdot W_c} \\
 &= \frac{(18.173) \cdot (0.3) \cdot (18.173 / 30) + (10.08) \cdot (0.7) \cdot (10.08 / 15)}{(57.87) \cdot (1.0)} \\
 &= 0.139
 \end{aligned}$$

These calculations illustrate those necessary for every iteration of the validation experiment to obtain Worthiness and cost figures.

B29972